APPENDIX A

Specification Volume 1

Specification of the Bluetooth System

Wireless connections made easy

Core

Bluetooth™

v1.0 B
December 1st 1999

BLUETOOTH DOC	Date / Day-Month-Year	N.B.	Document No.
	01 Dec 99		1.C.47/1.0 B
Responsible	e-mail address		Status

Bluetooth.

Specification of the Bluetooth System

Version 1.0 B

Bluetooth.

Revision History

The Revision History is shown in Appendix I on page 868

Contributors

The persons who contributed to this specification are listed in Appendix II on page 879.

Web Site

This specification can also be found on the Bluetooth website: http://www.bluetooth.com

Disclaimer and copyright notice

Bluetooth.

MASTER TABLE OF CONTENTS

For the Bluetooth Profiles, *See Volume 2.*

Part C **Volume 1 (1:2)**

LINK MANAGER PROTOCOL

Part D **Volume 1 (1:2)**

LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION

Part F:3 **Volume 1 (1:2)**

TELEPHONY CONTROL PROTOCOL SPECIFICATION

Part F:4 **Volume 1 (1:2)**

INTEROPERABILITY REQUIREMENTS FOR BLUETOOTH AS A WAP BEARER

Bluetooth.

Part H:1 **Volume 1 (2:2)**

HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

Bluetooth.

Bluetooth.

Appendix IX **Volume 1 (2:2)**

MESSAGE SEQUENCE CHARTS

Part F:1

RFCOMM with TS 07.10

Serial Port Emulation

This document specifies the RFCOMM protocol by specifying a subset of the ETSI TS 07.10 standard, along with some Bluetooth-specific adaptations

CONTENTS

1 INTRODUCTION

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. The protocol is based on the ETSI standard TS 07.10. This document does not contain a complete specification. Instead, references are made to the relevant parts of the TS 07.10 standard. Only a subset of the TS 07.10 standard is used, and some adaptations of the protocol are specified in this document.

1.1 OVERVIEW

RFCOMM is a simple transport protocol, with additional provisions for emulating the 9 circuits of RS-232 (EIATIA-232-E) serial ports.

The RFCOMM protocol supports up to 60 simultaneous connections between two BT devices. The number of connections that can be used simultaneously in a BT device is implementation-specific.

1.2 DEVICE TYPES

For the purposes of RFCOMM, a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them. Figure 1.1 shows the complete communication path. (In this context, the term *application* may mean other things than end-user application; e.g. higher layer protocols or other services acting on behalf of end-user applications.)

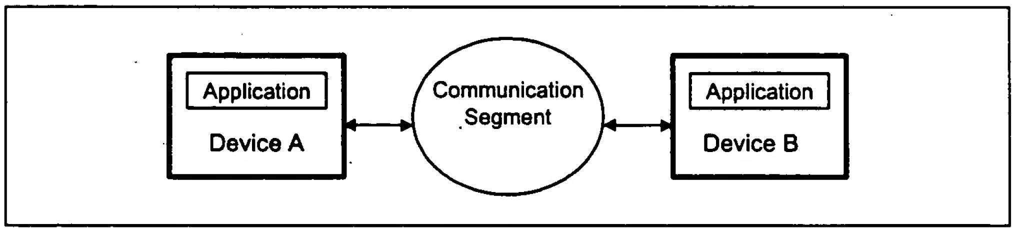

Figure 1.1: RFCOMM Communication Segment

RFCOMM is intended to cover applications that make use of the serial ports of the devices in which they reside. In the simple configuration, the communication segment is a BT link from one device to another (direct connect), see Figure 1.2. Where the communication segment is another network, BT is used for the path between the device and a network connection device like a modem. RFCOMM is only concerned with the connection between the devices in the direct connect case, or between the device and a modem in the network case. RFCOMM can support other configurations, such as modules that communicate via BT on one side and provide a wired interface on the other side, as shown in Figure 1.3. These devices are not really modems but offer a similar service. They are therefore not explicitly discussed here.

Basically two device types exist that RFCOMM must accommodate. Type 1 devices are communication end points such as computers and printers. Type 2 devices are those that are part of the communication segment; e.g. modems. Though RFCOMM does not make a distinction between these two device types in the protocol, accommodating both types of devices impacts the RFCOMM protocol.

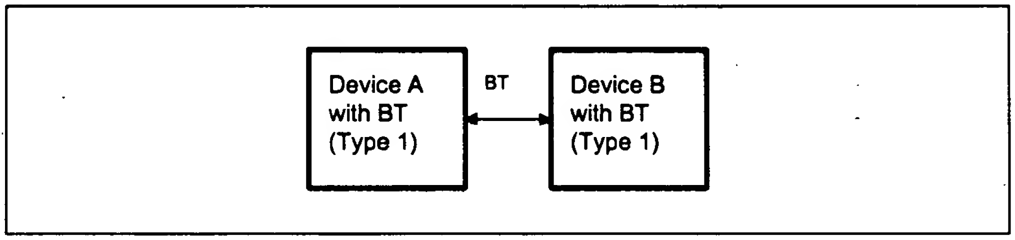

Figure 1.2: RFCOMM Direct Connect

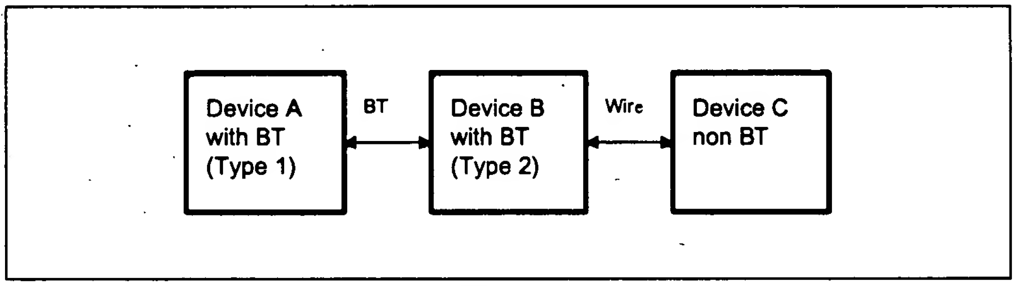

Figure 1.3: RFCOMM used with legacy COMM device

The information transferred between two RFCOMM entities has been defined to support both type 1 and type 2 devices. Some information is only needed by type 2 devices while other information is intended to be used by both. In the protocol, no distinction is made between type 1 and type 2. It is therefore up to the RFCOMM implementers to determine if the information passed in the RFCOMM protocol is of use to the implementation. Since the device is not aware of the type of the other device in the communication path, each must pass on all available information specified by the protocol.

1.3 BYTE ORDERING

This document uses the same byte ordering as the TS 07.10 specification; i.e. all binary numbers are in Least Significant Bit to Most Significant Bit order, reading from left to right.

2 RFCOMM SERVICE OVERVIEW

RFCOMM emulates RS-232 (EIATIA-232-E) serial ports. The emulation includes transfer of the state of the non-data circuits. RFCOMM has a built-in scheme for null modem emulation.

In the event that a baud rate is set for a particular port through the RFCOMM service interface, that will not affect the actual data throughput in RFCOMM; i.e. RFCOMM does not incur artificial rate limitation or pacing. However, if either device is a type 2 device (relays data onto other media), or if data pacing is done above the RFCOMM service interface in either or both ends, actual throughput will, on an average, reflect the baud rate setting.

RFCOMM supports emulation of multiple serial ports between two devices and also emulation of serial ports between multiple devices, see Section 2.3 on page 393.

2.1 RS-232 CONTROL SIGNALS

RFCOMM emulates the 9 circuits of an RS-232 interface. The circuits are listed below.

Pin	Circuit Name
102	Signal Common
103	Transmit Data (TD)
104	Received Data (RD)
105	Request to Send (RTS)
106	Clear to Send (CTS)
107	Data Set Ready (DSR)
108	Data Terminal Ready (DTR)
109	Data Carrier Detect (CD)
125	Ring Indicator (RI)

Table 2.1: Emulated RS-232 circuits in RFCOMM

2.2 NULL MODEM EMULATION

RFCOMM is based on TS 07.10. When it comes to transfer of the states of the non-data circuits, TS 07.10 does not distinguish between DTE and DCE devices. The RS-232 control signals are sent as a number of DTE/DCE independent signals, see Table 2.2.

TS 07.10 Signals	Corresponding RS-232 Control Signals
RTC	DSR, DTR
RTR	RTS, CTS
IC	RI
DV	DCD

Table 2.2: TS 07.10 Serial Port Control Signals

The way in which TS 07.10 transfers the RS-232 control signals creates an implicit null modem when two devices of the same kind are connected together. Figure 2.1 shows the null modem that is created when two DTE are connected via RFCOMM. No single null-modem cable wiring scheme works in all cases; however the null modem scheme provided in RFCOMM should work in most cases.

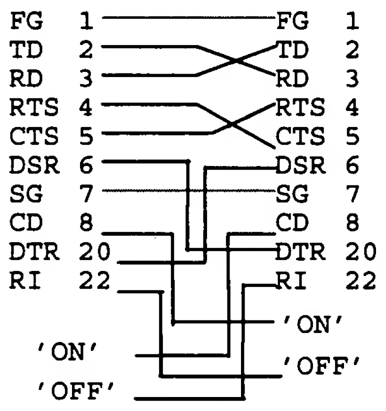

Figure 2.1: RFCOMM DTE–DTE Null Modem Emulation

2.3 MULTIPLE EMULATED SERIAL PORTS

2.3.1 Multiple Emulated Serial Ports between two Devices

Two BT devices using RFCOMM in their communication may open multiple emulated serial ports. RFCOMM supports up to 60 open emulated ports; however the number of ports that can be used in a device is implementation-specific.

A Data Link Connection Identifier (DLCI) [1] identifies an ongoing connection between a client and a server application. The DLCI is represented by 6 bits, but its usable value range is 2...61; in TS 07.10, DLCI 0 is the dedicated control channel, DLCI 1 is unusable due to the concept of Server Channels, and DLCI 62-63 is reserved. The DLCI is unique within one RFCOMM session between two devices. (This is explained further in Section 2.3.2) To account for the fact that both client and server applications may reside on both sides of an RFCOMM session, with clients on either side making connections independent of each other, the DLCI value space is divided between the two communicating devices using the concept of RFCOMM server channels. This is further described in Section 5.4.

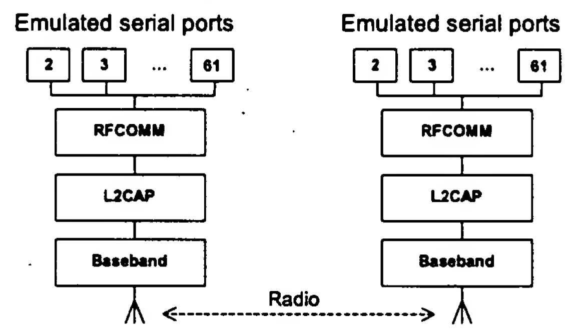

Figure 2.2: Multiple Emulated Serial Ports.

2.3.2 Multiple Emulated Serial Ports and Multiple BT Devices

If a BT device supports multiple emulated serial ports and the connections are allowed to have endpoints in different BT devices, then the RFCOMM entity must be able to run multiple TS 07.10 multiplexer sessions, see Figure 2.3. Note that each multiplexer session is using its own L2CAP channel ID (CID). The ability to run multiple sessions of the TS 07.10 multiplexer is optional for RFCOMM.

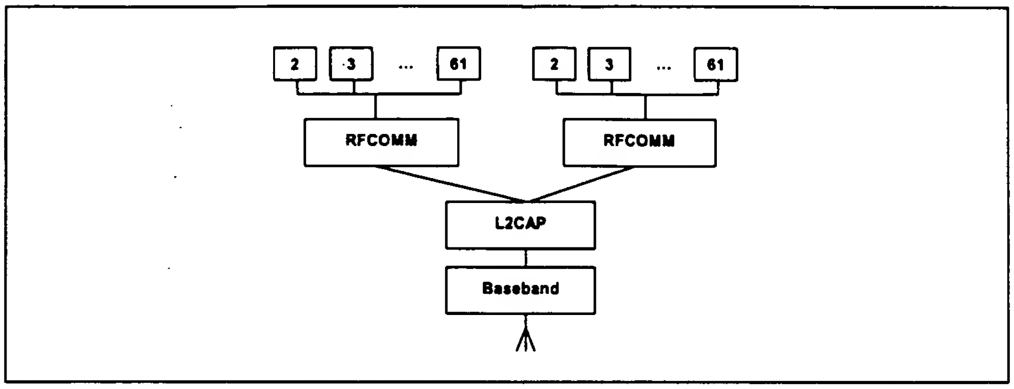

Figure 2.3: Emulating serial ports coming from two BT devices.

3 SERVICE INTERFACE DESCRIPTION

RFCOMM is intended to define a protocol that can be used to emulate serial ports. In most systems, RFCOMM will be part of a port driver which includes a serial port emulation entity.

3.1 SERVICE DEFINITION MODEL

The figure below shows a model of how RFCOMM fits into a typical system. This figure represents the RFCOMM reference model.

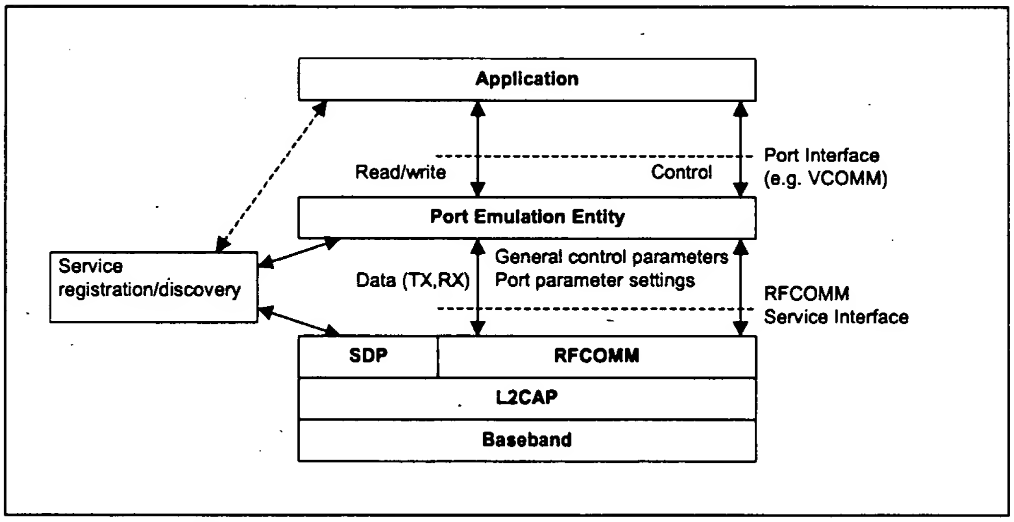

Figure 3.1: RFCOMM reference model

The elements of the RFCOMM reference model are described below.

Element	Description
Application	Applications that utilize a serial port communication interface
Port Emulation Entity	The port emulation entity maps a system-specific communication interface (API) to the RFCOMM services. The port emulation entity plus RFCOMM make up a port driver
RFCOMM	Provides a transparent data stream and control channel over an L2CAP channel. Multiplexes multiple emulated serial ports
Service Registration/ Discovery	Server applications register here on local device, and it provides services for client applications to discover how to reach server applications on other devices.
L2CAP	Protocol multiplexing, SAR
Baseband	Baseband protocols defined by BT

4 TS 07.10 SUBSET SUPPORTED BY RFCOMM

4.1 OPTIONS AND MODES

RFCOMM uses the basic option of TS 07.10.

4.2 FRAME TYPES

Table 4.1 shows the TS 7.10 frame types that are supported in RFCOMM.

Frame Types
Set Asynchronous Balanced Mode (SABM) command
Unnumbered Acknowledgement (UA) response
Disconnected Mode (DM) response
Disconnect (DISC) command
Unnumbered information with header check (UIH) command and response

Table 4.1: Supported frame types in RFCOMM

The 'Unnumbered Information (UI) command and response' are not supported by RFCOMM. Since the error recovery mode option of the TS 07.10 protocol is not used in RFCOMM none of the associated frame types are supported.

4.3 COMMANDS

TS 07.10 defines a multiplexer that has a dedicated control channel, DLCI 0. The control channel is used to convey information between two multiplexers. The following commands in TS 07.10 are supported by RFCOMM:

Supported Control Channel Commands
Test Command (Test)
Flow Control On Command (Fcon)
Flow Control Off Command (Fcoff)
Modem Status Command (MSC)
Remote Port Negotiation Command (RPN)
Remote Line Status (RLS)
DLC parameter negotiation (PN)
Non Supported Command Response (NSC)

Whenever a non-supported command type is received a 'Non-Supported Command Response (NSC)' should be sent.

4.4 CONVERGENCE LAYERS

RFCOMM only supports the type 1 convergence layer in TS 07.10.

The Modem Status Command (MSC) shall be used to convey the RS-232 control signals and the break signal for all emulated serial ports.

5 TS 07.10 ADAPTATIONS FOR RFCOMM

5.1 MEDIA ADAPTATION

The opening flag and the closing flags in the 07.10 basic option frame are not used in RFCOMM, instead it is only the fields contained between the flags that are exchanged between the L2CAP layer and RFCOMM layer, see Figure 5.1.

Flag	Address	Control	Length Indicator	Information	FCS	Flag
0111 1101	1 octet	1 octet	1 or 2 octets	Unspecified length but integral number of octets	1 octet	0111 1101

Figure 5.1: Frame Structure for Basic option. Note that the opening and closing flags from the 07.10 Basic option are excluded in RFCOMM.

5.1.1 FCS calculation

In 07.10, the frame check sequence (FCS) is calculated on different sets of fields for different frame types. These are the fields that the FCS are calculated on:

For SABM, DISC, UA, DM frames: on Address, Control and length field.

For UIH frames: on Address and Control field.

(This is stated here for clarification, and to set the standard for RFCOMM; the fields included in FCS calculation have actually changed in version 7.0.0 of TS 07.10, but RFCOMM will not change the FCS calculation scheme from the one above.)

5.2 TS 07.10 MULTIPLEXER START-UP AND CLOSEDOWN PROCEDURE

The start-up and closedown procedures as specified in section 5.7 in TS 07.10 are not supported. This means that the AT-command AT+CMUX is not supported by RFCOMM, neither is the multiplexer close down (CLD) command.

At any time, there must be at most one RFCOMM session between any pair of devices. When establishing a new DLC, the initiating entity must check if there already exists an RFCOMM session with the remote device, and if so, establish the new DLC on that. A session is identified by the Bluetooth BD_ADDR of the two endpoints[1].

5.2.1 Start-up procedure

The device opening up the first emulated serial port connection between two devices is responsible for first establishing the multiplexer control channel. This involves the following steps:

- Establish an L2CAP channel to the peer RFCOMM entity, using L2CAP service primitives, see L2CAP “Service Primitives” on page 295.
- Start the RFCOMM multiplexer by sending SABM command on DLCI 0, and await UA response from peer entity. (Further optional negotiation steps are possible.)

After these steps, DLCs for user data traffic can be established.

5.2.2 Close-down procedure

The device closing the last connection (DLC) on a particular session is responsible for closing the multiplexer by closing the corresponding L2CAP channel.

Closing the multiplexer by first sending a DISC command frame on DLCI 0 is optional, but it is mandatory to respond correctly to a DISC (with UA response).

5.2.3 Link loss handling

If an L2CAP link loss notification is received, the local RFCOMM entity is responsible for sending a connection loss notification to the port emulation/proxy entity for each active DLC. Then all resources associated with the RFCOMM session should be freed.

The appropriate action to take in the port emulation/proxy entity depends on the API on top. For example, for an emulated serial port (vCOMM), it would be suitable to drop CD, DSR and CTS signals (assuming device is a DTE).

1. This implies that, when responding to an L2CAP connection indication, the RFCOMM entity should save and associate the new RFCOMM session with the remote BD_ADDR. This is, at least, necessary if subsequent establishment of a DLC in the opposite direction is possible (which may depend on device capabilities).

5.3 SYSTEM PARAMETERS

Table 5.1 contains all the applicable system parameters for the RFCOMM implementation of the TS 07.10 multiplexer.

System Parameter	Value
Maximum Frame Size *(N1)*	Default: 127 (negotiable range 23 – 32767)
Acknowledgement Timer *(T1)*	60 seconds
Response Timer for Multiplexer Control Channel *(T2)*	60 seconds

Table 5.1: System parameter values

Note: The timer T1 is the timeout for *frames* sent with the P/F-bit set to one (this applies only to SABM and DISC frames in RFCOMM). T2 is the timeout for *commands* sent in UIH frames on DLCI 0.

Since RFCOMM relies on lower layers to provide reliable transmission, the default action performed on timeouts is to close down the multiplexer session. The only exception to this is when trying to set up a new DLC on an existing session; i.e. waiting for the UA response for a SABM command. In this case, the initiating side may defer the timeout by an unspecified amount of time if it has knowledge that the delay is due to user interaction (e.g. authentication procedure in progress). When/if the connection attempt is eventually considered to have timed out, the initiating side must send a DISC command frame on the same DLCI as the original SABM command frame, in order to notify the other party that the connection attempt is aborted. (After that the initiating side will, as usual, expect a UA response for the DISC command.)

5.4 DLCI ALLOCATION WITH RFCOMM SERVER CHANNELS

To account for the fact that both client and server applications may reside on both sides of an RFCOMM session, with clients on either side making connections independent of each other, the DLCI value space is divided between the two communicating devices using the concept of RFCOMM server channels and a direction bit.

The RFCOMM server channel number is a subset of the bits in the DLCI part of the address field in the TS 07.10 frame.

Bit No.	1	2	3	4	5	6	7	8
TS 07.10	EA	C/R	DLCI					
RFCOMM	EA	C/R	D	Server Channel				

Table 5.2: The format of the Address Field

Server applications registering with an RFCOMM service interface are assigned a Server Channel number in the range 1...30. [0 and 31 should not be used since the corresponding DLCIs are reserved in TS 07.10] It is this value that should be registered in the Service Discovery Database, see Section 7.2.

For an RFCOMM session, the initiating device is given the direction bit D=1 (and conversely, D=0 in the other device). When establishing a new data link connection on an existing RFCOMM session, the direction bit is used in conjunction with the Server Channel to determine the DLCI to use to connect to a specific application. This DLCI is thereafter used for all packets in both directions between the endpoints.

In effect, this partitions the DLCI value space such that server applications on the non-initiating device are reachable on DLCIs 2,4,6,...,60; and server applications on the initiating device are reachable on DLCIs 3,5,7,...,61. (Note that for a device that supports multiple simultaneous RFCOMM sessions to two or more devices, the direction bit might not be the same on all sessions.)

An RFCOMM entity making a new DLC on an existing session forms the DLCI by combining the Server Channel for the application on the other device, and the inverse of its own direction bit for the session.

DLCI 1 and 62-63 are reserved and never used in RFCOMM.

5.5 MULTIPLEXER CONTROL COMMANDS

Note that in TS 07.10, some Multiplexer Control commands pertaining to specific DLCIs may be exchanged on the control channel (DLCI 0) *before* the corresponding DLC has been established. (This refers the PN and RPN commands.) All such states associated with an individual DLC must be reset to their default values upon receiving a DISC command frame, or when closing the DLC from the local side. This is to ensure that all DLC (re-)establishments on the same session will have predictable results, irrespective of the session history.

5.5.1 Remote Port Negotiation Command (RPN)

The RPN command can be used before a new DLC is opened and should be used whenever the port settings change.

The RPN command is specified as optional in TS 07.10, but it is mandatory to recognize and respond to it in RFCOMM. (Although the handling of individual settings are implementation-dependent.)

5.5.2 Remote Line Status Command (RLS)

This command is used for indication of remote port line status.

The RLS command is specified as optional in TS 07.10, but it is mandatory to recognize and respond to it in RFCOMM. (Although the handling of individual settings are implementation-dependent.)

5.5.3 DLC parameter negotiation (PN)

The PN command is specified as optional in TS 07.10, but it is mandatory to recognize and respond to it in RFCOMM. This command can be used before a new DLC is opened.

There are some parameters in the PN command which convey information not applicable to RFCOMM. These fields must therefore be set to predetermined values by the sender, and they must be ignored by the receiver. This concern the following fields (see table 3 in ref. [1]):

- I1-I4 must be set to 0. (Meaning: use UIH frames.)
- CL1-CL4 must be set to 0. (Meaning: use convergence layer type 1.)
- T1-T8 must be set to 0. (Meaning: acknowledgment timer *T1*, which is not negotiable in RFCOMM.)
- NA1-NA8 must be set to 0. (Meaning: number of retransmissions *N2*; always 0 for RFCOMM)
- K1-K3 must be set to 0. (Meaning: defines the window size for error recovery mode, which is not used for RFCOMM.)

If a command is received with invalid (or for some reason unacceptable) values in any field, a DLC parameter negotiation response must be issued with values that are acceptable to the responding device.

6 FLOW CONTROL

Wired ports commonly use flow control such as RTS/CTS to control communications. On the other hand, the flow control between RFCOMM and the lower layer L2CAP depends on the service interface supported by the implementation. In addition RFCOMM has its own flow control mechanisms. This section describes the different flow control mechanisms.

6.1 L2CAP FLOW CONTROL IN OVERVIEW

L2CAP relies on the flow control mechanism provided by the Link Manager layer in the baseband. The flow control mechanism between the L2CAP and RFCOMM layers is implementation-specific.

6.2 WIRED SERIAL PORT FLOW CONTROL

Wired Serial ports falls into two camps – software flow control using characters such as XON/XOFF, and flow control using RTS/CTS or DTR/DSR circuits. These methods may be used by both sides of a wired link, or may be used only in one direction.

6.3 RFCOMM FLOW CONTROL

The RFCOMM protocol provides two flow control mechanisms:

1. The RFCOMM protocol contains flow control commands that operate on the aggregate data flow between two RFCOMM entities; i.e. all DLCIs are affected. The control channel commands, FCon and FCoff, are defined in section 5.4.6.3 in ref [1].
2. The Modem Status command as defined in section 5.4.6.3 in ref [1] is the flow control mechanism that operates on individual DLCI.

6.4 PORT EMULATION ENTITY SERIAL FLOW CONTROL

On Type 1 devices some port drivers (Port Emulation Entities plus RFCOMM) will need to provide flow control services as specified by the API they are emulating. An application may request a particular flow control mechanism like XON/XOFF or RTS/CTS and expect the port driver to handle the flow control. On type 2 devices the port driver may need to perform flow control on the non-RFCOMM part of the communication path; i.e. the physical RS-232 port. This flow control is specified via the control parameters sent by the peer RFCOMM entity (usually a type 1 device). The description of flow control in this section is for port drivers on type 1 devices.

Since RFCOMM already has its own flow control mechanism, the port driver does not need to perform flow control using the methods requested by the application. In the ideal case, the application sets a flow control mechanism and assumes that the COMM system will handle the details. The port driver could then simply ignore the request and rely on RFCOMM's flow control. The application is able to send and receive data, and does not know or care that the port driver did not perform flow control using the mechanism requested. However, in the real world some problems arise.

- The RFCOMM-based port driver is running on top of a packet-based protocol where data may be buffered somewhere in the communication path. Thus, the port driver cannot perform flow control with the same precision as in the wired case.
- The application may decide to apply the flow control mechanism itself in addition to requesting flow control from the port driver.

These problems suggest that the port driver must do some additional work to perform flow control emulation properly. Here are the basic rules for flow control emulation.

- The port driver will not solely rely on the mechanism requested by the application but use a combination of flow control mechanisms.
- The port driver must be aware of the flow control mechanisms requested by the application and behave like the wired case when it sees changes on the non-data circuits (hardware flow control) or flow control characters in the incoming data (software flow control). For example, if XOFF and XON characters would have been stripped in the wired case they must be stripped by the RFCOMM based port driver.
- If the application sets a flow control mechanism via the port driver interface and then proceeds to invoke the mechanism on its own, the port driver must behave in a manner similar to that of the wired case (e.g. If XOFF and XON characters would have been passed through to the wire in the wired case the port driver must also pass these characters).

These basic rules are applied to emulate each of the wired flow control schemes. Note that multiple types of flow control can be set at the same time. Section 5.4.8 in ref [1] defines each flow control mechanism.

7 INTERACTION WITH OTHER ENTITIES

7.1 PORT EMULATION AND PORT PROXY ENTITIES

This section defines how the RFCOMM protocol should be used to emulate serial ports. Figure 7.1 shows the two device types that the RFCOMM protocol supports.

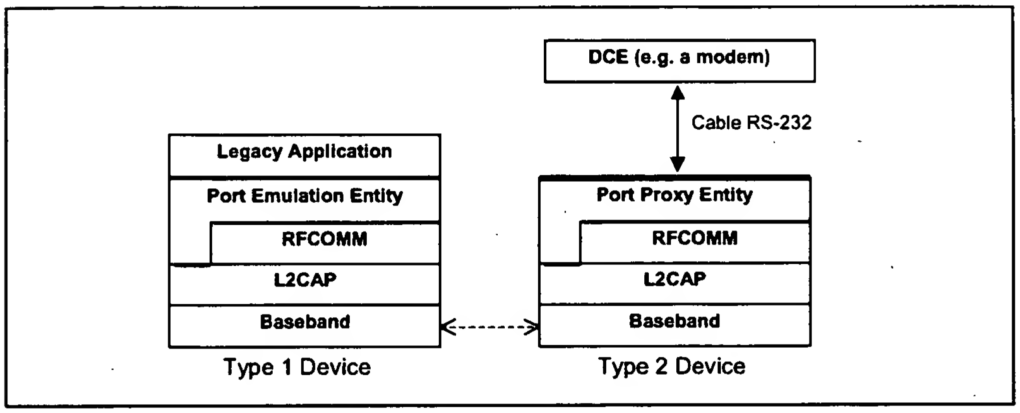

Figure 7.1: The RFCOMM communication model

Type 1 devices are communication endpoints such as computers and printers. Type 2 devices are part of a communication segment; e.g. modems.

7.1.1 Port Emulation Entity

The port emulation entity maps a system specific communication interface (API) to the RFCOMM services.

7.1.2 Port Proxy Entity

The port proxy entity relays data from RFCOMM to an external RS-232 interface linked to a DCE. The communications parameters of the RS-232 interface are set according to received RPN commands, see Section 5.5.1.

7.2 SERVICE REGISTRATION AND DISCOVERY

Registration of individual applications or services, along with the information needed to reach those (i.e. the RFCOMM Server Channel) is the responsibility of each application respectively (or possibly a Bluetooth configuration application acting on behalf of legacy applications not directly aware of Bluetooth).

Below is a template/example for developing service records for a given service or profile using RFCOMM. It illustrates the inclusion of the ServiceClassList with a single service class, a ProtocolDescriptor List with two protocols (although there may be more protocols on top of RFCOMM). The example shows the use of one other universal attribute (ServiceName). For each service running on top of RFCOMM, appropriate SDP-defined universal attributes and/or service-specific attributes will apply. For additional information on Service Records, see the SDP Specification, Section 2.2 on page 332.

The attributes that a client application needs (at a minimum) to connect to a service on top of RFCOMM are the ServiceClassIDList and the ProtocolDescriptorList (corresponding to the shaded rows in the table below).

Item	Definition	Type/Size	Value	Attribute ID
ServiceClassIDList			Note1	0x0001
ServiceClass0	Note5	UUID/32-bit	Note1	
ProtocolDescriptorList				0x0004
Protocol0	L2CAP	UUID/32-bit	L2CAP /Note1	
Protocol1	RFCOMM	UUID/32-bit	RFCOMM /Note1	
ProtocolSpecificParameter0	Server Channel	Uint8	N = server channel #	
[other protocols]		UUID/32-bit	Note1	
[other protocol-specific parameters]	Note3	Note3	Note3	
ServiceName	Displayable text name	DataElement/ String	'Example service'	Note2
[other universal attributes as appropriate for this service]	Note4	Note4	Note4	Note4
[service-specific attributes]	Note3	Note3	Note3	Note3

Notes:

1. Defined in "Bluetooth Assigned Numbers" on page 1009.
2. For national language support for all 'displayable' text string attributes, an offset has to be added to the LanguageBaseAttributeIDList value for the selected language (see the SDP Specification, Section 5.1.14 on page 365 for details).
3. To be defined (where necessary) for the specific service.
4. For a specific service some of the SDP-defined universal attributes may apply. See the SDP Specification, Section 5.1 on page 358.
5. This indicates the class of service. It can be a single entry or a list of service classes ranging from generic to most specific.

7.3 LOWER LAYER DEPENDENCIES

7.3.1 Reliability

RFCOMM uses the services of L2CAP to establish L2CAP channels to RFCOMM entities on other devices. An L2CAP channel is used for the RFCOMM/TS 07.10 multiplexer session. On such a channel, the TS 07.10 frames listed in Section 4.2 are sent, with the adaptation defined in Section 5.1.

Some frame types (SABM and DISC) as well as UIH frames with multiplexer control commands sent on DLCI 0 always require a response from the remote entity, so they are acknowledged on the RFCOMM level (but not retransmitted in the absence of acknowledgment, see Section 5.3). Data frames do not require any response in the RFCOMM protocol, and are thus unacknowledged.

Therefore, RFCOMM must require L2CAP to provide channels with maximum reliability, to ensure that all frames are delivered in order, and without duplicates. Should an L2CAP channel fail to provide this, RFCOMM expects a link loss notification, which should be handled by RFCOMM as described in Section 5.2.3.

7.3.2 Low power modes

If all L2CAP channels towards a certain device are idle for a certain amount of time, a decision may be made to put that device in a low power mode (i.e. use hold, sniff or park, see 'Baseband Specification' Section 10.10.3 on page 125). This will be done without any interference from RFCOMM. RFCOMM can state its latency requirements to L2CAP.This information may be used by lower layers to decide which low power mode(s) to use.

The RFCOMM protocol as such does not suffer from latency delays incurred by low power modes, and consequentially, this specification does not state any maximum latency requirement on RFCOMM's behalf. Latency sensitivity inherently depends on application requirements, which suggests that an RFCOMM service interface implementation could include a way for applications to state latency requirements, to be aggregated and conveyed to L2CAP by the RFCOMM implementation. (That is if such procedures make sense for a particular platform.)

8 REFERENCES

[1] TS 07.10, ver 6.3.0, ETSI

[2] Bluetooth L2CAP Specification

[3] Bluetooth SDP Specification

[4] Bluetooth Assigned Numbers

9 TERMS AND ABBREVIATIONS

The following terms are used throughout the document.

DTE	Data Terminal Equipment – in serial communications, DTE refers to a device at the endpoint of the communications path; typically a computer or terminal
DCE	Data Circuit-Terminating Equipment – in serial communications, DCE refers to a device between the communication endpoints whose sole task is to facilitate the communications process; typically a modem
RFCOMM initiator	The device initiating the RFCOMM session; i.e.setting up RFCOMM channel on L2CAP and starting RFCOMM multiplexing with the SABM command frame on DLCI 0 (zero)
RFCOMM Client	An RFCOMM client is an application that requests a connection to another application (RFCOMM server)
RFCOMM Server	An RFCOMM server is an application that awaits a connection from an RFCOMM client on another device. What happens after such a connection is established is not within the scope of this definition
RFCOMM Server Channel	This is a subfield of the TS 07.10 DLCI number. This abstraction is used to allow both server and client applications to reside on both sides of an RFCOMM session

Part F:2

IrDA INTEROPERABILITY

The IrOBEX protocol is utilized by the Bluetooth technology. In Bluetooth, OBEX offers the same features for applications as within the IrDA protocol hierarchy, enabling the applications to work over the Bluetooth protocol stack as well as the IrDA stack.

CONTENTS

1 INTRODUCTION

The goal of this document is to enable the development of application programs that function well over both short-range RF and IR media. Each media type has its advantages and disadvantages but the goal is for applications to work over both. Rather than fragment the application domain, this document defines the intersection point where Bluetooth and IrDA applications may converge. That intersection point is IrOBEX [1].

IrOBEX is a session protocol defined by IrDA. This protocol is now also utilized by the Bluetooth technology, making it possible for applications to use either the Bluetooth radio technology or the IrDA IR technology. However, even though both IrDA and Bluetooth are designed for short-range wireless communications, they have some fundamental differences relating to the lower-layer protocols. IrOBEX will therefore be mapped over the lower layer protocols which are adopted by Bluetooth.

This document defines how IrOBEX (OBEX for short) is mapped over RFCOMM [2] and TCP/IP [3]. Originally, OBEX (Object Exchange Protocol) was developed to exchange data objects over an infrared link and was placed within the IrDA protocol hierarchy. However, it can appear above other transport layers, now RFCOMM and TCP/IP. At this moment, it is worth mentioning that the OBEX over TCP/IP implementation is an optional feature for Bluetooth devices supporting the OBEX protocol.

The IrOBEX specification [1] provides a model for representing objects and a session protocol, which structures the dialogue between two devices. The IrOBEX protocol follows a client/server **request-response** paradigm for the conversation format.

Bluetooth uses only the connection-oriented OBEX even though IrDA has specified the connectionless OBEX also. The reasons for the connection-oriented approach are:

- OBEX is mapped over the connection-oriented protocols in the Bluetooth architecture.
- Most of application profiles using OBEX and Bluetooth needs a connection-oriented OBEX to provide the functionality described for the features included in these profiles.
- The connectionless OBEX with the connection-oriented one would raise the interoperability problems, which are not desirable.

1.1 OBEX AND BLUETOOTH ARCHITECTURE

Figure 1.1 depicts part of the hierarchy of the Bluetooth architecture and shows the placement of the OBEX protocol and the application profiles using it (See also Section 5 on page 425). The protocols can also communicate with the service discovery DB even though the figure does not show it.

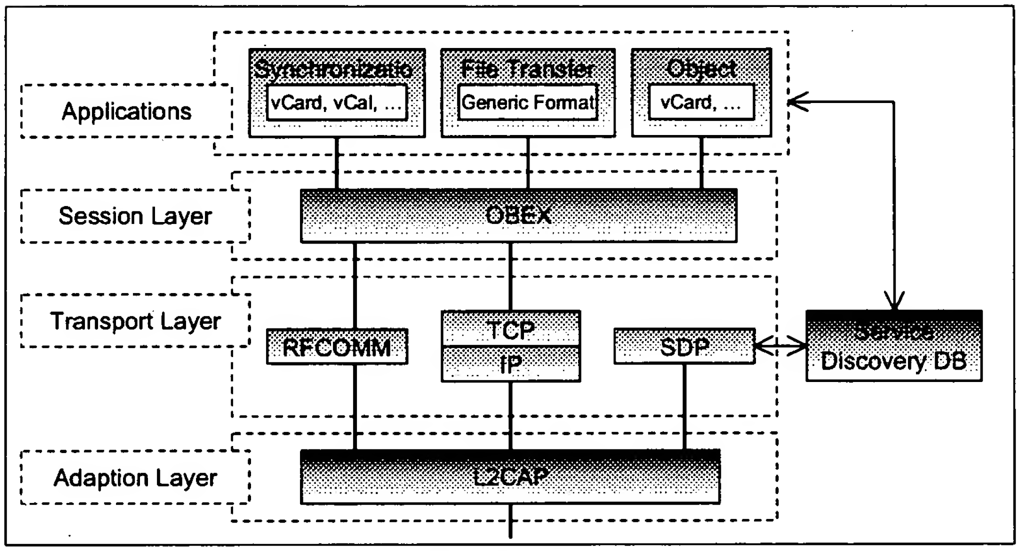

Figure 1.1: Part of Bluetooth Protocol Hierarchy

In the Bluetooth system, the purpose of the OBEX protocol is to enable the exchange of data objects. The typical example could be an object push of business cards to someone else. A more complex example is synchronizing calendars on multiple devices using OBEX. Also, the File Transfer applications can be implemented using OBEX. For the Object Push and Synchronization applications, content formats can be the vCard [4], vCalendar [5], vMessage [6], and vNotes [6] formats. The vCard, vCalendar, vMessage, and vNotes describe the formats for the electronic business card, the electronic calendaring and scheduling, the electronic message and mails, and the electronic notes, respectively.

1.2 BLUETOOTH OBEX-RELATED SPECIFICATIONS

Bluetooth Specification includes five separate specifications related to OBEX and applications using it:

1. Bluetooth IrDA Interoperability Specification (This specification)

- Defines how the applications can function over both Bluetooth and IrDA
- Specifies how OBEX is mapped over RFCOMM and TCP
- Defines the application profiles using OBEX over Bluetooth

2. Bluetooth Generic Object Exchange Profile Specification [7]
- Generic interoperability specification for the application profiles using OBEX
- Defines the interoperability requirements of the lower protocol layers (e.g. Baseband and LMP) for the application profiles

3. Bluetooth Synchronization Profile Specification [8]
- Application Profile for the Synchronization applications
- Defines the interoperability requirements for the applications within the Synchronization application profile
- Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.

4. Bluetooth File Transfer Profile Specification [9]
- Application Profile for the File Transfer applications
- Defines the interoperability requirements for the applications within the File Transfer application profile.
- Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.

5. Bluetooth Object Push Profile Specification [10]
- Application Profile for the Object Push applications
- Defines the interoperability requirements for the applications within the Object Push application profile.
- Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.

1.3 OTHER IROBEX IMPLEMENTATIONS

Over IR, OBEX has also been implemented over IrCOMM and Tiny TP. The Bluetooth technology does not define these protocols as transport protocols for OBEX, but they can be supported by independent software vendors if desired.

2 OBEX OBJECT AND PROTOCOL

This section is dedicated to the model of OBEX objects and the OBEX session protocol. The section is intended to be read with the IrOBEX specification [1].

2.1 OBJECT

The OBEX object model (Section 2 in [1]) describes how OBEX objects are presented. The OBEX protocol can transfer an object by using the **Put**- and **Get**-operations (See Section 2.2.3 and 2.2.4). One object can be exchanged in one or more **Put**-requests or **Get**-responses.

The model handles both information about the object (e.g. type) and object itself. It is composed of headers, which consist of a header ID and value (See Section 2.1 in [1]). The header ID describes what the header contains and how it is formatted, and the header value consists of one or more bytes in the format and meaning specified by Header ID. The specified headers are **Count**, **Name**, **Type**, **Length**, **Time**, **Description**, **Target**, **HTTP**, **Body**, **End of Body**, **Who**, **Connection ID**, **Application Parameters**, **Authenticate Challenge**, **Authenticate Response**, **Object Class**, and User-Defined Headers. These are explained in detail by Section 2.2 in the IrOBEX specification.

2.2 SESSION PROTOCOL

The OBEX operations are formed by **response-request** pairs. Requests are issued by the client and responses by the server. After sending a request, the client waits for a response from the server before issuing a new request. Each request packet consists of a one-byte opcode (See Section 3.3 in [1]), a two-byte length indicator, and required or optional data depending on the operation. Each response packet consists of a one-byte response code (See Section 3.2.1 in [1]), a two-byte length indicator, and required or optional data depending on the operation.

In the following subsections, the OBEX operations are explained in general.

2.2.1 Connect Operation

An OBEX session is started, when an application asks the first time to transmit an OBEX object. An OBEX client starts the establishment of an OBEX connection. The session is started by sending a **Connect**-request (See Section 3.3.1 in [1]). The request format is:

Byte 0	Bytes 1 and 2	Byte 3	Byte 4	Bytes 5 and 6	Byte 7 to n
0x80 (opcode)	Connect request packet length	OBEX version number	Flags	Maximum OBEX packet length	Optional headers

Note. The Big Endian format is used to define the byte ordering for the PDUs (requests and responses) in this specification as well as in the IrOBEX specification; i.e. the most significant byte (MSB) is always on left and the least significant byte (LSB) on right.

At the remote host, the **Connect**-request is received by the OBEX server, if it exists. The server accepts the connection by sending the successful response to the client. Sending any other response (i.e. a non-successful response) back to the client indicates a failure to make a connection. The response format is:

Byte 0	Bytes 1 and 2	Byte 3	Byte 4	Bytes 5 and 6	Byte 7 to n
Response code	Connect response packet length	OBEX version number	Flags	Maximum OBEX packet length	Optional headers

The response codes are list in the Section 3.2.1 in the IrOBEX specification. The bytes 5 and 6 define the maximum OBEX packet length, which can be received by the server. This value may differ from the length, which can be received by the client. These **Connect**-request and response packets must each fit in a single packet.

Once a connection is established it remains 'alive', and is only disconnected by requests/responses or by failures (i.e. the connection is not automatically disconnected after each OBEX object has completely transmitted).

2.2.2 Disconnect Operation

The disconnection of an OBEX session occurs when an application, which is needed for an OBEX connection, is closed or the application wants to change the host to which the requests are issued. The client issues the **Disconnect**-request (See Section 3.3.2 in [1]) to the server. The request format is:

Byte 0	Bytes 1 and 2	Byte 3
0x81	Packet length	Optional headers

The request cannot be refused by the server. Thus, it has to send the response, and the response format is:

Byte 0	Bytes 1 and 2	Byte 3
0xA0	Response packet length	Optional response headers

2.2.3 Put Operation

When the connection has been established between the client and server the client is able to push OBEX objects to the server. The **Put**-request is used to push an OBEX object (See Section 3.3.3 in [1]). The request has the following format.

Byte 0	Bytes 1 and 2	Byte 3
0x02 (0x82 when Final bit set)	Packet length	Sequence of headers

A **Put**-request consists of one or more request packets, depending on how large the transferred object is, and how large the packet size is. A response packet from the server is required for every **Put**-request packet. Thus, one response is not permitted for several request packets, although they consist of one OBEX object. The response format is:

Byte 0	Bytes 1 and 2	Byte 3
Response code	Response packet length	Optional response headers

2.2.4 Get Operation

When the connection has been established between the client and server, the client is also able to pull OBEX objects from the server. The **Get**-request is used to pull an OBEX object (See Section 3.3.4 in [1]). The request has the following format.

Byte 0	Bytes 1 and 2	Byte 3
0x03 (0x83 when Final bit set)	Packet length	Sequence of headers starting with Name

The object is returned as a sequence of headers, and the client has to send a request packet for every response packet. The response format is:

Byte 0	Bytes 1 and 2	Byte 3
Response code	Response packet length	Optional response headers

2.2.5 Other Operations

Other OBEX operations consist of a **SetPath**-, and an **Abort**-operation. These are carefully explained in the Sections 3.3.5-6 in the IrOBEX specification. It is important to note that the client can send an **Abort**-request after each response – even in the middle of a request/response sequence. Thus, the whole OBEX object does <u>not</u> have to be received before sending an **Abort**-request. In addition to these operations, the IrOBEX specification facilitates user-defined operations, but their use may not necessarily be adopted in Bluetooth.

3 OBEX OVER RFCOMM

This section specifies how OBEX is mapped over RFCOMM, which is the multiplexing and transport protocol based on ETSI TS 07.10 [11] and which also provides a support for serial cable emulation. The Bluetooth devices supporting the OBEX protocol must satisfy the following requirements.

1. The device supporting OBEX must be able to function as either a client, a server, or both
2. All servers running simultaneously on a device must use separate RFCOMM server channels
3. Applications (service/server) using OBEX must be able to register the proper information into the service discovery database. This information for different application profiles is specified in the profile specifications

3.1 OBEX SERVER START-UP ON RFCOMM

When a client sends a connecting request, a server is assumed to be ready to receive requests. However, before the server is ready to receive (i.e. is running) certain prerequisites must be fulfilled before the server can enter the listening mode:

1. The server must open an RFCOMM server channel
2. The server must register its capabilities into the service discovery database

After this, other hosts are able to find the server if needed, and the server listens for get requests from clients.

3.2 RECEIVING OBEX PACKETS FROM SERIAL PORT

As discussed earlier, one object can be exchanged over one or more **Put**-requests or **Get**-responses (i.e. the object is received in one or several packets). However, if OBEX is running directly over the serial port, it does not receive packets from RFCOMM. Instead, a byte stream is received by OBEX from a serial port emulated by RFCOMM.

To detect packets in the byte stream, OBEX has to look for opcodes or response codes (See Chapter 2.2) depending on whether a packet is a request or a response. The opcodes and response code can be thought of as the start flags of packets. In OBEX packets, there is no ‘end flag’ that would indicate the end of a packet. However, after the opcode or response code, the length of a packet is received in the next two bytes. Thus, the whole length of a packet is known, and the boundary of two packets can be determined.

All data that is not recognized must be dumped. This could cause a synchronization problem but, considering the nature of the OBEX protocol, this is not a problem over RFCOMM, which provides reliable transport over Bluetooth.

3.3 CONNECTION ESTABLISHMENT

A client initiates the establishment of a connection. However, the following sequence of tasks must occur before the client is able to send the first request for data:

1. By using the SD protocol described in the SDP specification [12], the client must discover the proper information (e.g. RFCOMM channel) associated with the server on which the connection can be established
2. The client uses the discovered RFCOMM channel to establish the RFCOMM connection
3. The client sends the **Connect**-request to the server, to establish an OBEX session. The session is established correctly if the client receives a successful response from the server

3.4 DISCONNECTION

The disconnection of an OBEX session over RFCOMM is straightforward. The disconnection is done by using the **Disconnect**-request (See Section 2.2.2). When the client has received the response, the next operation is to close the RFCOMM channel assigned to the OBEX client.

3.5 PUSHING AND PULLING OBEX PACKETS OVER RFCOMM

Data is pushed in OBEX packets over RFCOMM by using **Put**-requests (See Section 2.2.3). After each request, a response is required before the next request with the data can be pushed.

Pulling data from a remote host happens by sending a **Get**-request (See Section 2.2.4. The data arrives in OBEX response packets. After each response, a new request has to be sent, to pull more data.

4 OBEX OVER TCP/IP

This section specifies how OBEX is mapped over the TCP/IP creating reliable connection-oriented services for OBEX. This specification does not define how TCP/IP is mapped over Bluetooth.

The Bluetooth devices, which support the OBEX protocol over TCP/IP, must satisfy the following requirements:

1. The device supporting OBEX must be able to function as either a client, or a server, or both
2. For the server, the TCP port number 650 is assigned by IANA. If an assigned number is not desirable, the port number can be a value above 1023. However, the use of the TCP port number (650) defined by IANA is highly recommended. The 0-1023 range is reserved by IANA (See [13])
3. The client must use a port number (on the client side), which is not within the 0-1023 range
4. Applications (service/server) using OBEX must be able to register the proper information into the service discovery database. This information for different application profiles is specified in the profile specifications

4.1 OBEX SERVER START-UP ON TCP/IP

When a client sends a **Put**- or **Get**-request, a server is assumed to be ready to receive requests. However, when the server is ready (i.e. is running), certain prerequisites must be fulfilled before the server can enter the listening mode:

1. The server must initialize a TCP port with the value 650 or value above 1023
2. The server registers its capabilities into the service discovery database

After this, other devices are able to find the server if needed, and the server listens for get requests from clients.

4.2 CONNECTION ESTABLISHMENT

A client initiates a connection. However, the following sequence of tasks must occur before a connection can be established:

1. By using, the SD protocol described in the SDP specification [12], the client discovers the proper information (e.g. TCP port number) associated with the server, to enable the connection can be established
2. The client initializes a socket associated to a TCP port number above 1023, and establishes a TCP connection with the host of the server
3. The client sends the **Connect**-request to the server, to establish an OBEX session. The session is established correctly if the client receives a successful response from the server.

4.3 DISCONNECTION

The disconnection of an OBEX session over TCP is straightforward. The disconnection is done by using the **Disconnect**-request (See Section 2.2.2). When the client has received the response, the next operation is to close the TCP port dedicated for this session.

4.4 PUSHING AND PULLING OBEX PACKETS OVER TCP

See Section 3.5.

5 BLUETOOTH APPLICATION PROFILES USING OBEX

Bluetooth SIG (Special Interest Group) has defined three separate application profiles using OBEX. These profiles are briefly introduced in this section.

5.1 SYNCHRONIZATION

Basically, the synchronization means comparing two object stores, determining their inequalities, and then unifying these two object stores. The Bluetooth devices supporting the synchronization may be desktop PCs, notebooks, PDAs, cellular phones, or smart phones.

The Bluetooth Synchronization profile uses the servers and clients compliant to the IrMC synchronization specified by IrDA (See Section 5 in [6]).
The Bluetooth Synchronization servers and clients must support the level 4 synchronization functionality specified in the IrMC specification.

The actual logic of the synchronization engines which process the synchronization algorithm at the client device is implementation-specific. It is therefore left to the participating software vendors, and is not considered in the Bluetooth specifications.

The synchronization is not limited to one type of application. The Bluetooth synchronization (i.e. the IrMC synchronization) enables four different application classes:

1. Phone Book – provides a means for a user to manage contact records
2. Calendar – enables a user to manage calendar items, and can also be used for 'to-do' or task lists
3. Messaging – lets a user manage messages (e.g. e-mails)
4. Notes – provides a means for a user to manage small notes

The interoperability requirements for the Bluetooth Synchronization profile are defined in the Synchronization Profile [8] and Generic Object Exchange Profile [7] specifications.

5.2 FILE TRANSFER

At the minimum, the File Transfer profile is intended for sending and retrieving generic files to and from the Bluetooth device. The File Transfer service also facilitates the browsing of the remote Bluetooth device's folder.

The interoperability requirements for the Bluetooth File Transfer profile are defined in the File Transfer Profile [9] and Generic Object Exchange Profile [7] specifications.

5.3 OBJECT PUSH

The Object Push profile is the special case of the File Transfer Profile for beaming objects and optionally pulling the default objects. At a minimum, it offers the capability to exchange business cards, but is not limited to this service.

The interoperability requirements for the Object Push profile are defined in the Object Push Profile [10] and Generic Object Exchange Profile [7] specifications.

6 REFERENCES

[1] Infrared Data Association, IrDA Object Exchange Protocol (IrOBEX), Version 1.2, April 1999

[2] Bluetooth RFCOMM with TS 07.10, on page 385

[3] Internet Engineering Task Force, IETF Directory List of RFCs (http://www.ietf.org/rfc/), May 1999.

[4] The Internet Mail Consortium, vCard - The Electronic Business Card Exchange Format, Version 2.1, September 1996.

[5] The Internet Mail Consortium, vCalendar - The Electronic Calendaring and Scheduling Exchange Format, Version 1.0, September 1996.

[6] Infrared Data Association, IrMC (Ir Mobile Communications) Specification, Version 1.1, February 1999.

[7] Bluetooth Generic Object Exchange Profile, see Volume 2.

[8] Bluetooth Synchronization Profile, see Volume 2.

[9] Bluetooth File Transfer Profile, see Volume 2.

[10] Bluetooth Object Push Profile, see Volume 2.

[11] ETSI, TS 07.10, Version 6.3.0

[12] Bluetooth Service Discovery Protocol, see Volume 2.

[13] Internet Assigned Numbers Authority, IANA Protocol/Number Assignments Directory (http://www.iana.org/numbers.html), May 1999.

7 LIST OF ACRONYMS AND ABBREVIATIONS

Abbreviation or Acronym	**Meaning**
GEOP	Generic Object Exchange Profile
IrDA	Infrared Data Association
IrMC	Ir Mobile Communications
L2CAP	Logical Link Control and Adaptation Protocol
LSB	Least Significant Byte
MSB	Most Significant Byte
OBEX	Object exchange protocol
PDU	Protocol Data Unit
RFCOMM	Serial cable emulation protocol based on ETSI TS 07.10
SD	Service Discovery
SDP	Service Discovery Protocol
SDDB	Service Discovery Database
TCP/IP	Transport Control Protocol/Internet Protocol

Part F:3

TELEPHONY CONTROL PROTOCOL SPECIFICATION

TCS Binary

This document describes the Bluetooth Telephony Control protocol Specification – Binary (TCS *Binary*), using a bit-oriented protocol. This protocol defines the call control signalling for the establishment of speech and data calls between Bluetooth devices.
In addition, it defines mobility management procedures for handling Bluetooth TCS devices.

CONTENTS

Bluetooth.

1 GENERAL DESCRIPTION

1.1 OVERVIEW

The Bluetooth Telephony Control protocol Specification Binary (TCS *Binary*) is based on the ITU-T Recommendation Q.931[1], applying the symmetrical provisions as stated in Annex D of Q.931. The resulting text does not discriminate between user and network side, but merely between Outgoing Side (the party originating the call) and Incoming Side (the party terminating the call). Effort was made to only apply those changes necessary for Bluetooth and foreseen applications, enabling re-use of Q.931 to the largest extent possible.

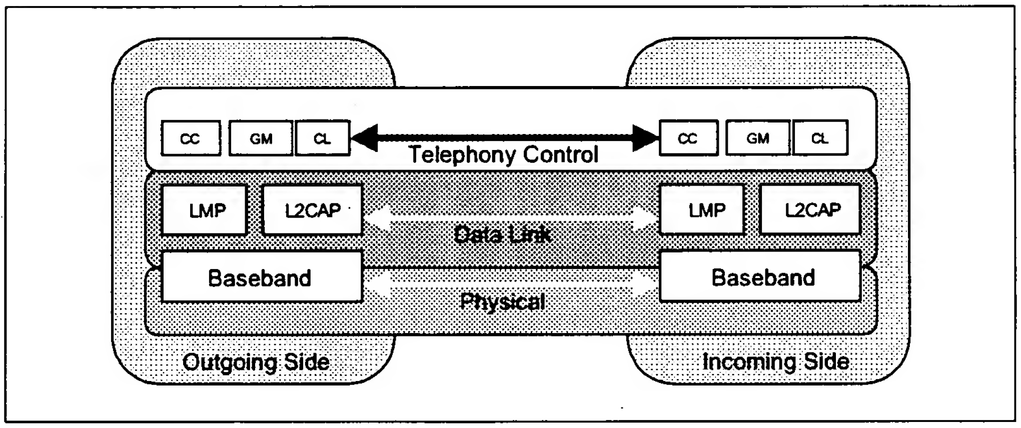

Figure 1.1: TCS within the Bluetooth stack

The TCS contains the following functionality:

- Call Control (CC) – signalling for the establishment and release of speech and data calls between Bluetooth devices
- Group Management – signalling to ease the handling of groups of Bluetooth devices
- ConnectionLess TCS (CL) – provisions to exchange signalling information not related to an ongoing call

1.2 OPERATION BETWEEN DEVICES

TCS uses point-to-point signalling and may use point-to-multipoint signalling. Point-to-point signalling is used when it is known to which side (Bluetooth device) a call needs to be established (*single-point configuration*).

Point-to-multipoint signalling may be used when there are more sides available for call establishment (*multi-point configuration*); e.g. when, for an incoming call, a home base station needs to alert all phones in range.

Point-to-point signalling is mapped towards a connection-oriented L2CAP channel, whereas point-to-multipoint signalling is mapped towards the connectionless L2CAP channel, which in turn is sent as broadcast information on the beacon channel (piconet broadcast).

Figure 1.2 illustrates point-to-point signalling to establish a voice or data call in a single-point configuration. First the other device is notified of the call request using the point-to-point signalling channel (A). Next, this signalling channel is used to further establish the speech or data channel (B).

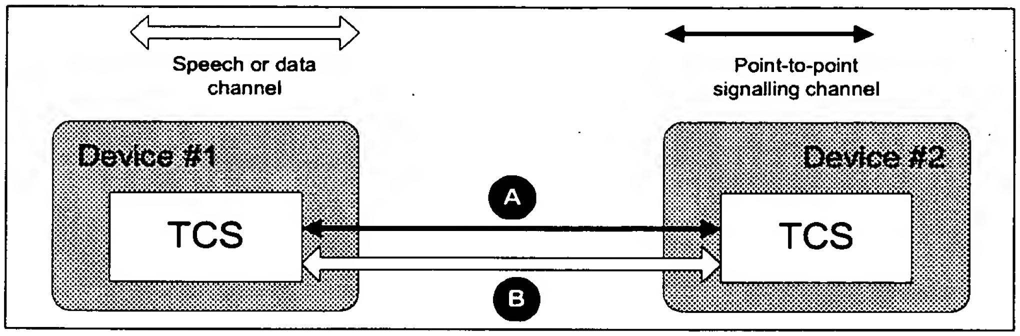

Figure 1.2: Point-to-point signalling in a single-point configuration

Figure 1.3 below illustrates how point-to-multipoint signalling and point-to-point signalling is used to establish a voice or data call in a multi-point configuration. First all devices are notified of the call request using point-to-multipoint signalling channel (A). Next, one of the devices answers the call on the point-to-point signalling channel (B); this signalling channel is used to further establish the speech or data channel (C).

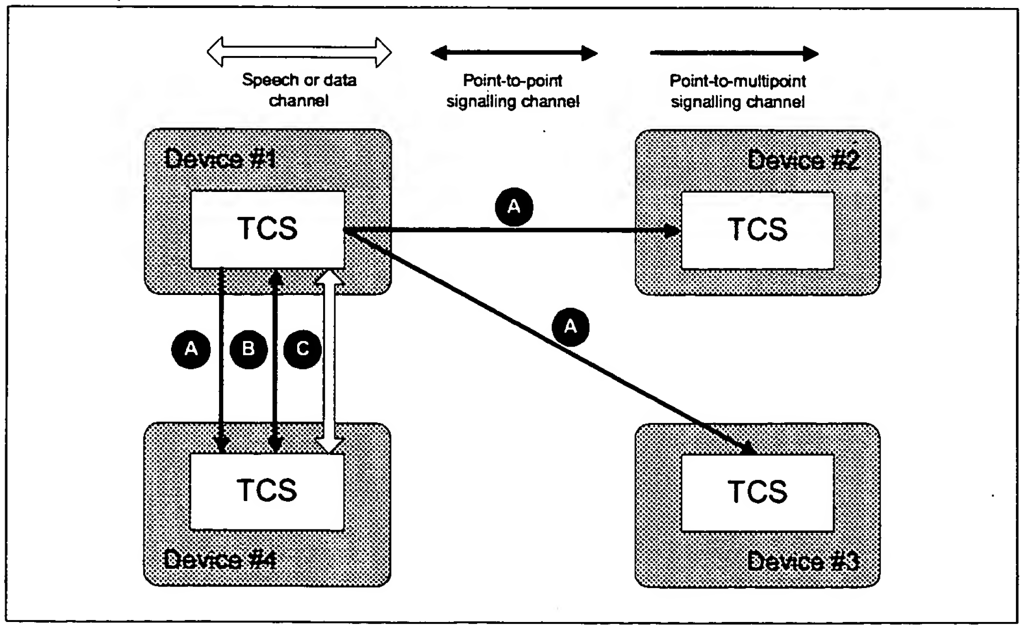

Figure 1.3: Signalling in a multi-point configuration

1.3 OPERATION BETWEEN LAYERS

TCS implementations should follow the general architecture described below (note that, for simplicity, handling of data calls is not drawn).

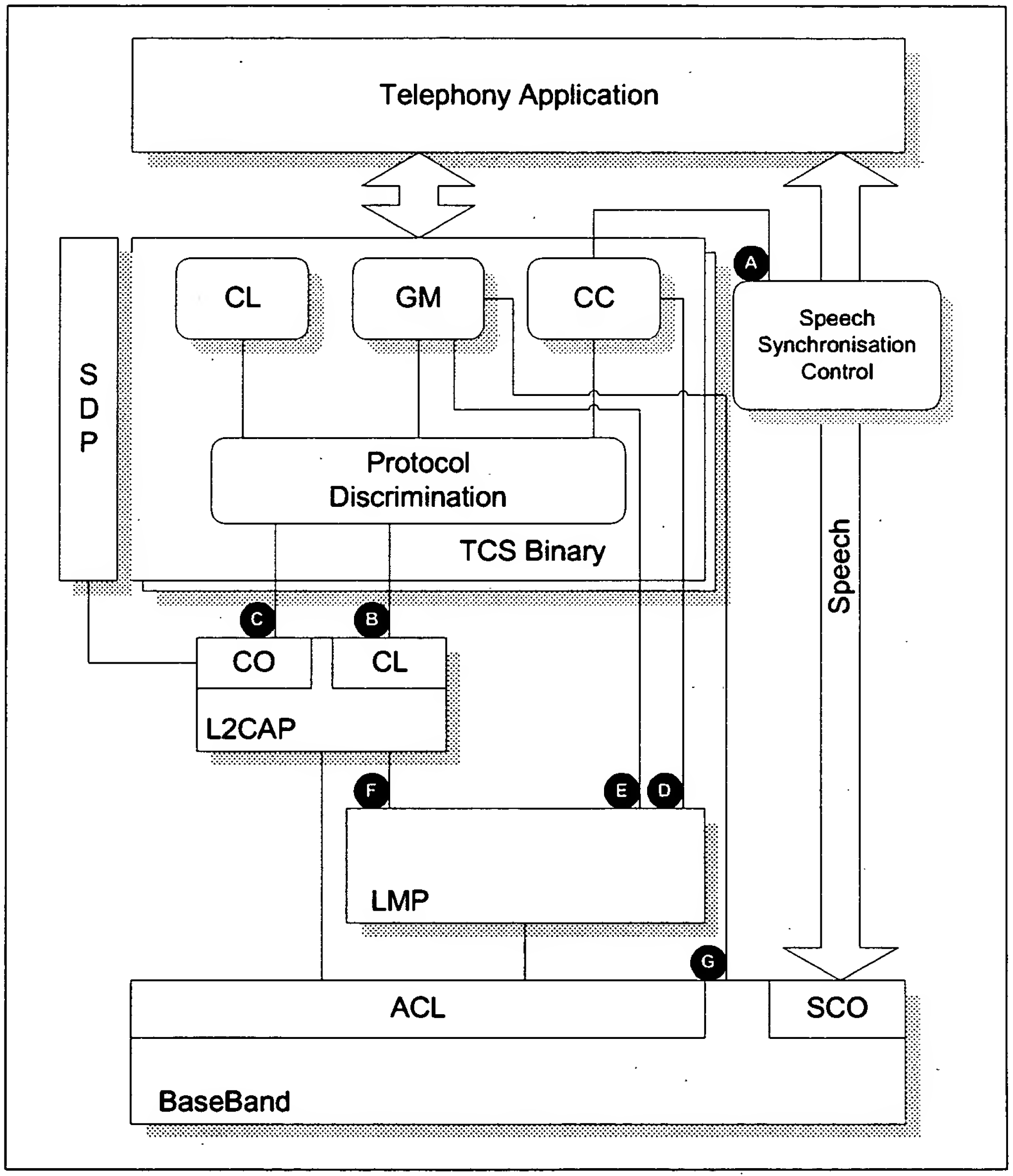

Figure 1.4: TCS Architecture

The internal structure of TCS Binary contains the functional entities Call Control, Group Management and ConnectionLess as described in Section 1.1 on page 435, complemented with the Protocol Discrimination which, based upon the TCS internal protocol discriminator, routes traffic to the appropriate functional entity.

To handle more calls simultaneously, multiple instances of TCS Binary may exist at the same time. Discrimination between the multiple instances can be based on the L2CAP channel identifier.

TCS Binary interfaces with a number of other (Bluetooth) entities to provide its (telephone) services to the application. The interfaces are identified in Figure 1.4 above, and information is exchanged across these interfaces for the following purposes:

A The Call Control entity provides information to the speech synchronization control about when to connect (disconnect) the speech paths. This information is based upon the call control messages (e.g. reception of CONNECT ACKNOWLEDGE or DISCONNECT, see Section 2 on page 439)

B To send a SETUP message (see Section 2.2.1 on page 439) using point-to-multipoint signalling, it is delivered on this interface to L2CAP for transmission on the connectionless channel. The other way round – L2CAP uses this interface to inform TCS of a SETUP message received on the connectionless channel. The connectionless L2CAP channel maps onto the piconet broadcast

C Whenever a TCS message needs to be sent using point-to-point signalling, it is delivered on this interface to L2CAP for transmission on a connection-oriented channel. During L2CAP channel establishment specific quality of service to be used for the connection will be indicated, in particular the usage of low power modes (L2CAP will inform LMP about this – interface F)

D The Call Control entity controls the LMP directly, for the purpose of establishing and releasing SCO links

E & G. The Group Management entity controls the LMP and LC/Baseband directly during initialization procedures to control (for example) the inquiry, paging and pairing.

2 CALL CONTROL (CC)

2.1 CALL STATES

The call states used by the TCS are those identified in Q.931[1], for the user side only. To allow for implementation within computing power- and memory-restricted devices, only a subset of the states is mandatory for TCS based implementations. This mandatory subset is termed **Lean TCS**.

The states are named as follows. States in bold are mandatory states, part of Lean TCS:

General States
- ***Null (0)***
- ***Active (10)***
- ***Disconnect request (11)***
- ***Disconnect indication (12)***
- ***Release request (19)***

Outgoing Side States
- ***Call initiated (1)***
- *Overlap sending (2)*
- *Outgoing call proceeding (3)*
- *Call delivered (4)*

Incoming Side States
- ***Call present (6)***
- *Call received (7)*
- ***Connect request (8)***
- *Incoming call proceeding (9)*
- *Overlap receiving (25)*

These states, together with the state transitions, have been indicated in the state diagram contained in Appendix 1 – TCS Call States. For clarity, a separate state diagram has been included for Lean TCS.

2.2 CALL ESTABLISHMENT

A connection-oriented L2CAP channel between the Outgoing and Incoming Side shall be available before any of the CC procedures can operate.

Additionally, in a multi-point configuration (see Section 1.2 on page 435), a connectionless L2CAP channel shall be available between the Outgoing and Incoming Side.

2.2.1 Call Request

The Outgoing Side initiates call establishment by sending a SETUP message, and starting timer T303.

In case of a single-point configuration (see Section 1.2 on page 435), the SETUP message is delivered on the connection-oriented channel.

In case of a multi-point configuration (see Section 1.2 on page 435), the SETUP message may be delivered on the connection-less channel. This causes the SETUP message to be transmitted as a broadcast message at every beacon instant (as described in Baseband Specification Section 10.8.4 on page 115).

If no response (as prescribed in Section 2.2.4 on page 441) is received from the Incoming Side before timer T303 expires, the Outgoing Side shall:

1. If the SETUP message was delivered on a connection-less channel, return to the Null state. This stops the transmission of the SETUP message.
2. If the SETUP message was delivered on a connection-oriented channel, send a RELEASE COMPLETE message to the Incoming Side. This message should contain cause # 102, *recovery on timer expiry.*

The SETUP message shall always contain the call class. It shall also contain all the information required by the Incoming Side to process the call. The number digits within the Called party number information element may optionally be incomplete, thus requiring the use of overlap sending (Section 2.2.3 on page 441). The SETUP message may optionally contain the Sending complete information element in order to indicate that the number is complete.

Following the transmission of the SETUP message, the Outgoing Side shall enter the Call initiated state. On receipt of the SETUP message the Incoming Side shall enter the Call present state.

2.2.2 Bearer selection

The SETUP message sent during the Call Request may contain the Bearer capability information element, to indicate the requested bearer. The Incoming Side may negotiate on the requested bearer by including a Bearer capability information element in the first message in response to the SETUP message.

The Bearer capability information element indicates which lower layer resources (the *bearer channel*) are used during a call. If bearer capability 'Synchronous Connection-Oriented (SCO)' is indicated, an SCO link will be used, with the indicated packet type and voice coding to enable speech calls. If bearer capability 'Asynchronous Connection-Less (ACL)' is indicated, an ACL link will be used. On top of this, there will be an L2CAP channel with indicated QoS requirements, to enable data calls. If bearer capability 'None' is indicated, no separate bearer channel will be established.

Note: it is the responsibility of the implementation to assure that the bearer capability as indicated is available to the call.

2.2.3 Overlap Sending

If the received SETUP message does not contain a Sending complete indication information element, and contains either –

a) incomplete called-number information, or

b) called-number information which the Incoming Side cannot determine to be complete,

then the Incoming Side shall start timer T302, send a SETUP ACKNOWLEDGE message to the Outgoing Side, and enter the Overlap receiving state.

When the SETUP ACKNOWLEDGE message is received, the Outgoing Side shall enter the Overlap sending state, stop timer T303, and start timer T304.

After receiving the SETUP ACKNOWLEDGE message, the Outgoing Side shall send the remainder of the call information (if any) in the called party number information element of one or more INFORMATION messages.

The Outgoing Side shall restart timer T304 when each INFORMATION message is sent.

The INFORMATION message, which completes the information sending, may contain a sending complete information element. The Incoming Side shall restart timer T302 on receipt of every INFORMATION message not containing a sending complete indication, if it cannot determine that the called party number is complete.

At the expiry of timer T304, the Outgoing Side shall initiate call clearing in accordance with Section 2.3.1 with cause #102, *recovery on timer expiry.*

At the expiry of timer T302, the Incoming Side shall:

- if it determines that the call information is incomplete, initiate call clearing in accordance with Section 2.3.1 with cause #28, *invalid number format.*
- otherwise the Incoming Side shall reply with a CALL PROCEEDING, ALERTING or CONNECT message.

2.2.4 Call Proceeding

2.2.4.1 Call proceeding, enbloc sending

If enbloc sending is used (i.e. the Incoming Side can determine it has received sufficient information in the SETUP message from the Outgoing Side to establish the call) the Incoming Side shall send a CALL PROCEEDING message to the Outgoing Side to acknowledge the SETUP message and to indicate that the call is being processed. Upon receipt of the CALL PROCEEDING message, the Outgoing Side shall enter the Outgoing Call proceeding state stop timer T303 and start timer T310. After sending the CALL PROCEEDING message, the Incoming Side shall enter the Incoming Call proceeding state.

2.2.4.2 Call proceeding, overlap sending

Following the occurrence of one of these conditions –

- the receipt by the Incoming Side of a Sending complete indication, or
- analysis by the Incoming Side that all call information necessary to effect call establishment has been received,

the Incoming Side shall send a CALL PROCEEDING message to the Outgoing Side, stop timer T302, and enter the Incoming Call proceeding state.

When the Outgoing Side receives of the CALL PROCEEDING message it shall enter the Outgoing Call proceeding state, stop timer T304 and, if applicable, start timer T310.

2.2.4.3 Expiry of timer T310

On expiry of T310 (i.e. if the Outgoing Side does not receive an ALERTING, CONNECT, DISCONNECT or PROGRESS message), the Outgoing Side shall initiate call clearing in accordance with Section 2.3.1 on page 446 with cause #102, *recovery on timer expiry.*

2.2.5 Call Confirmation

Upon receiving an indication that user alerting has been initiated at the called address, the Incoming Side shall send an ALERTING message, and shall enter the Call received state.

When the Outgoing Side receives the ALERTING message, the Outgoing Side may begin an internally generated alerting indication and shall enter the Call delivered state. The Outgoing Side shall stop timer T304 (in case of overlap receiving), stop timer T303 or T310 (if running), and start timer T301 (unless another internal altering supervision timer function exists).

On expiry of T301, the Outgoing Side shall initiate call clearing in accordance with Section 2.3.1 on page 446 with cause #102, *recovery on timer expiry.*

2.2.6 Call Connection

An Incoming Side indicates acceptance of an incoming call by sending a CONNECT message to the Outgoing Side, and stopping the user alerting. Upon sending the CONNECT message the Incoming Side shall start timer T313.

On receipt of the CONNECT message, the Outgoing Side shall stop any internally generated alerting indications, shall stop (if running) timers T301, T303, T304, and T310, shall complete the requested bearer channel to the Incoming Side, shall send a CONNECT ACKNOWLEDGE message, and shall enter the Active state.

The CONNECT ACKNOWLEDGE message indicates completion of the requested bearer channel. Upon receipt of the CONNECT ACKNOWLEDGE message, the Incoming Side shall connect to the bearer channel, stop timer T313 and enter the Active state.

When timer T313 expires prior to the receipt of a CONNECT ACKNOWLEDGE message, the Incoming Side shall initiate call clearing in accordance with Section 2.3.1 on page 446 with cause #102, *recovery on timer expiry.*

2.2.7 Call Information

While in the Active state, both sides may exchange any information related to the ongoing call using INFORMATION messages.

2.2.8 Non-selected user clearing

When the call has been delivered on a connection-less channel (in case of a multi-point configuration), in addition to sending a CONNECT ACKNOWLEDGE message to the Incoming Side selected for the call, the Outgoing Side shall send a RELEASE message (indicating cause #26, *non-selected user clearing*) to all other Incoming Sides that have sent SETUP ACKNOWLEDGE, CALL PROCEEDING, ALERTING, or CONNECT messages in response to the SETUP message. These RELEASE messages are used to notify the Incoming Sides that the call is no longer offered to them.

2.2.9 In-band tones and announcements

When the Incoming Side provides in-band tones/announcements, and if the requested bearer implies speech call, the Incoming Side will first complete the bearer channel (if not already available). Then a progress indicator #8, *in-band information or appropriate pattern is now available* is sent simultaneously with the application of the in-band tone/announcement. This progress indicator may be included in any call control message that is allowed to contain the progress indicator information element or, if there is no call state change, in a dedicated PROGRESS message.

Upon receipt of this message, the Outgoing Side may connect (if not already connected) to the bearer channel to receive the in-band tone/announcement.

2.2.10 Failure of call establishment

In the Call present, Overlap receiving, Incoming call proceeding, or Call received states, the Incoming Side may initiate clearing as described in Section 2.3 on page 446 with a cause value indicated. Examples of some the cause values that may be used to clear the call, when the Incoming Side is in the Call present, Overlap receiving, or Incoming call proceeding state are the following:

#1 unassigned (unallocated) number
#3 no route to destination
#17 user busy
#18 no user responding
#22 number changed
#28 invalid number format (incomplete number)
#34 no circuit/channel available
#44 requested circuit/channel not available
#58 bearer capability not presently available
#65 bearer capability not implemented

Examples of two of the cause values that may be used to clear the call when the Incoming Side is in the Call received state are as follows:

#19 no answer from user (user alerted)
#21 call rejected by user

2.2.11 Call Establishment Message Flow

The figure below provides a complete view of the messages exchanged during successful Call Establishment, as described in the sections above. The mandatory messages, part of the Lean TCS, are indicated by a solid arrow. A dotted arrow indicates the optional messages. A triangle indicates a running timer.

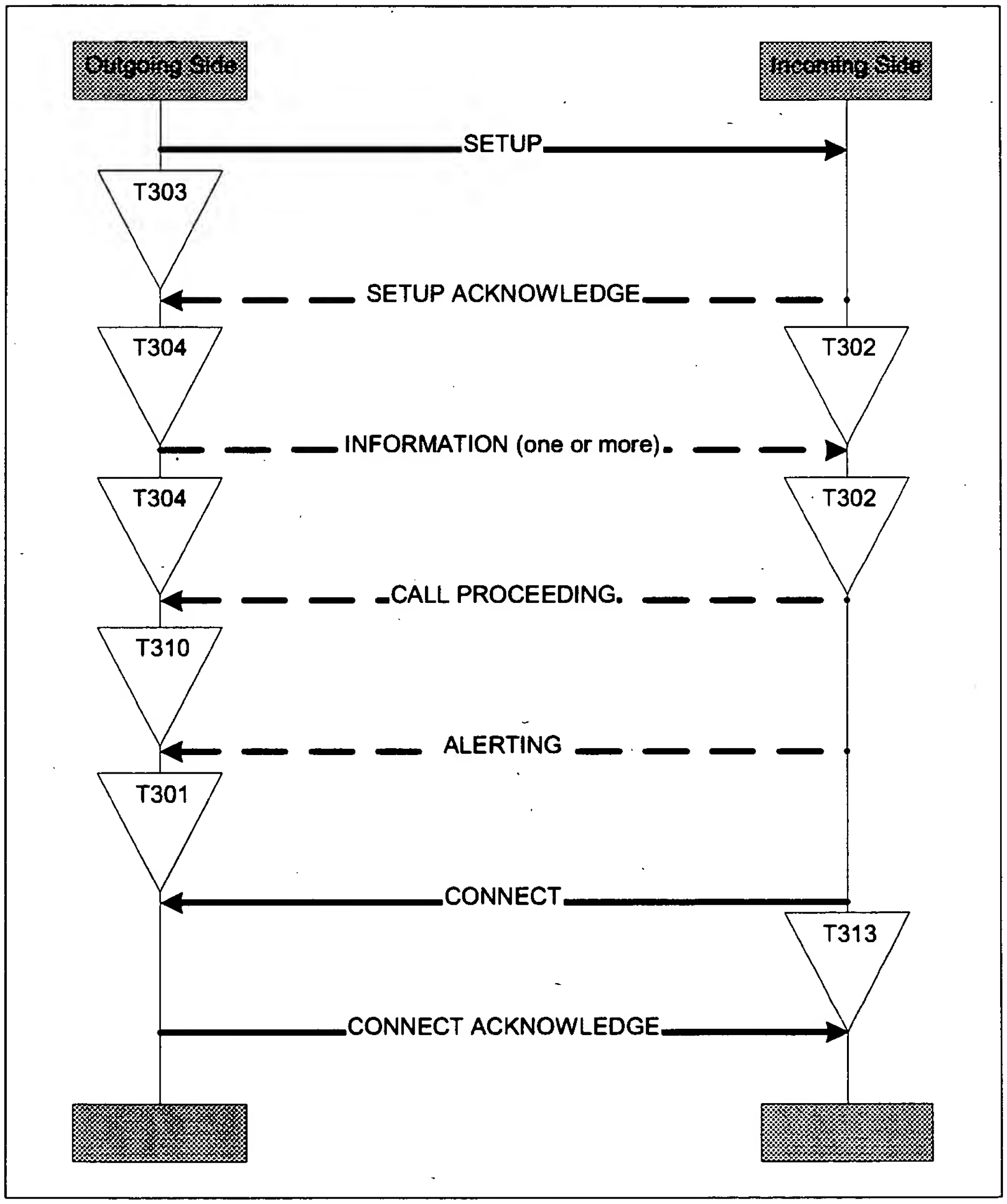

Figure 2.1: Call establishment message flow

2.3 CALL CLEARING

2.3.1 Normal Call Clearing

Apart from the exceptions identified in Section 2.3.2 on page 447, the clearing procedures are symmetrical and may be initiated by either the Outgoing or the Incoming Side. In the interest of clarity, the following procedures describe only the case where the Outgoing Side initiates clearing.

On sending or receiving any call clearing message, any protocol timer other than T305 and T308 shall be stopped.

The Outgoing Side shall initiate clearing by sending a DISCONNECT message, starting timer T305, disconnecting from the bearer channel, and entering the Disconnect request state.

The Incoming Side shall enter the Disconnect indication state upon receipt of a DISCONNECT message. This message prompts the Incoming Side to disconnect from the bearer channel. Once the channel used for the call has been disconnected, the Incoming Side shall send a RELEASE message to the Outgoing Side, start timer T308, and enter the Release request state.

On receipt of the RELEASE message the Outgoing Side shall cancel timer T305, release the bearer channel, send a RELEASE COMPLETE message, and return to the Null state.

Following the receipt of a RELEASE COMPLETE message from the Outgoing Side, the Incoming Side shall stop timer T308, release the bearer channel, and return to the Null state.

If the Outgoing Side does not receive a RELEASE message in response to the DISCONNECT message before timer T305 expires, it shall send a RELEASE message to the Incoming Side with the cause number originally contained in the DISCONNECT message, start timer T308 and enter the Release request state.

If in the Release request state, a RELEASE COMPLETE message is not received before timer T308 expires, the side that expected the message shall return to the Null state.

Clearing by the called user employing user-provided tones/announcements

In addition to the procedures described above, if the requested bearer signals a speech call, the Outgoing Side may apply in-band tones/announcements in the clearing phase. When in-band tones/announcements are provided, the Outgoing Side will first complete the bearer channel (if not already available), and next send the DISCONNECT message containing progress indicator *#8, in-band information or appropriate pattern is now available.*

Upon receipt of this message, the Incoming Side may connect (if not already connected) to the bearer channel to receive the in-band tone/announcement, and enter the Disconnect indication state.

The Incoming Side may subsequently continue clearing (before the receipt of a RELEASE from the Outgoing Side) by disconnecting from the bearer channel, sending a RELEASE message, starting timer T308, and entering the Release request state.

2.3.2 Abnormal Call Clearing

Under normal conditions, call clearing is initiated when either side sends a DISCONNECT message and follows the procedures defined in Section 2.3.1 on page 446. The only exceptions to the above rule are as follows:

a In response to a SETUP message, the Incoming Side can reject a call (e.g. because of unavailability of suitable resources) by responding with a RELEASE COMPLETE message provided no other response has previously been sent, and enter the Null state

b In case of a multi-point configuration, non-selected user call clearing will be initiated with RELEASE message(s) from the Outgoing Side (Section 2.2.8 on page 443)

c In case of a multi-point configuration, where the SETUP message is delivered on an connection-less channel, if a remote (calling) user disconnect indication is received during call establishment, any Incoming Side which has responded, or subsequently responds, shall be cleared by a RELEASE message, and the procedures of Section 2.3.1 on page 446 are then followed for that user. The Outgoing Side enters the Null state upon completion of clearing procedures for all responding Incoming Sides.

2.3.3 Clear Collision

Clear collision occurs when the Incoming and the Outgoing Sides simultaneously transfer DISCONNECT messages. When either side receives a DISCONNECT message while in the Disconnect request state, the side shall stop timer T305, disconnect the bearer channel (if not disconnected), send a RELEASE message, start timer T308, and enter the Release request state.

Clear collision can also occur when both sides simultaneously transfer RELEASE messages. The entity receiving such a RELEASE message while within the Release request state shall stop timer T308, release the bearer channel, and enter the Null state (without sending or receiving a RELEASE COMPLETE message).

2.3.4 Call Clearing Message Flow

The figure below provides the complete view on the messages exchanged during normal Call Clearing, as described in the sections above. All messages are mandatory.

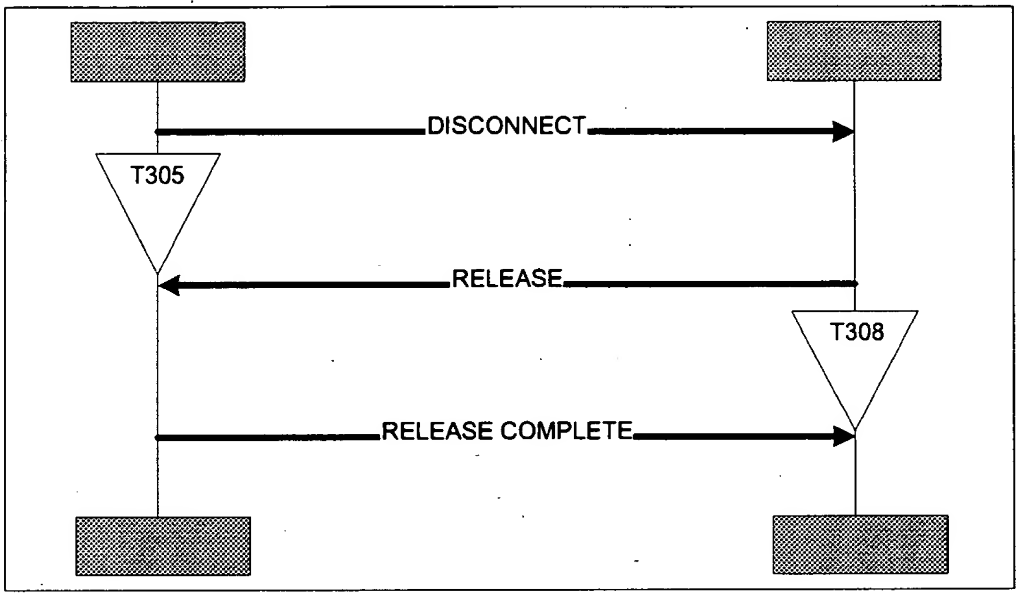

Figure 2.2: Call clearing message flow

3 GROUP MANAGEMENT (GM)

3.1 OVERVIEW

The Group Management entity provides procedures for managing a group of devices.

The following procedures are supported:

- Obtain access rights (Section 3.3 on page 451)
  enables the requesting device to use the telephony services of another device, part of a group of devices
- Configuration distribution (Section 3.4 on page 452)
  facilitates the handling and operation of a group of devices
- Fast inter-member access (Section 3.5 on page 453)
  enables faster contact establishment between devices of the same group

A connection-oriented L2CAP channel between devices shall be available before any of the GM procedures can operate.

For group management, the concept of Wireless User Group (WUG) is used.

3.2 THE WIRELESS USER GROUP

3.2.1 Description

A WUG consists of a number of Bluetooth units supporting TCS. One of the devices is called the WUG master. The WUG master is typically a gateway, providing the other Bluetooth devices – called WUG members – with access to an external network. All members of the WUG in range are members of a piconet (active or parked). Master of this piconet is always the WUG master.

The main relational characteristics of a WUG are:

- All units that are part of a WUG know which unit is the WUG master and which other units are member of this WUG. WUG members receive this information from the WUG master.
- When a new unit has paired with the WUG master, it is able to communicate and perform authentication and encryption with any other unit part of the WUG without any further pairing/initialization. The WUG master provides the required authentication and encryption parameters to the WUG members.

Both relational characteristics are maintained through the Configuration distribution procedure.

3.2.2 Encryption within the WUG

In order to allow for encrypted transmission on the connectionless L2CAP channel, the WUG master issues a temporary key (K_{master}). As a Bluetooth unit is not capable of switching between two or more encryption keys in real time, this key is normally also used for encrypted transmission on the connection-oriented channel (individually addressed traffic). Since the WUG master piconet may be in operation for extended periods without interruption, the K_{master} shall be changed periodically.

In order to allow for authentication and encryption to be performed between WUG members, the WUG master may use the Configuration distribution procedure to issue link keys that the WUG members use for communication with each other. Just as if pairing had created these keys, the keys are unique to a pair of WUG members and hence a WUG member uses a different key for every other WUG member it connects to.

The Configuration distribution shall always be performed using encrypted links. The K_{master} shall not be used for encryption; rather the WUG master shall ensure that the semi-permanent key for the specific WUG member addressed shall be used.

3.2.3 Unconscious pairing

For TCS, pairing a device with the WUG master implies pairing a device with all members of the WUG.This is achieved using the Configuration distribution procedure. This avoids the user of the device having to pair with each and every device of the WUG individually.

In Bluetooth, pairing is not related to a specific service but rather to a specific device. After pairing, all services provided by a device are accessible, if no further application- or device-specific protection is provided.

Without further provisions, pairing a device with the WUG master implies that all services provided by the new device are accessible to all other WUG members. And vice versa, without further provisions, the new device can access all services provided by other WUG members.

For this reason, implementers of TCS – and in particular the Configuration distribution procedure – are recommended to add provisions where:

1. a new device entering the WUG is not mandated to initiate the Obtain access rights procedure to become a WUG member, and is consequently only able to use the services provided by the WUG master (gateway)
2. a WUG master can reject a request to obtain access rights
3. a WUG member is not forced to accept the pairing information received during the Configuration distribution

This applies in particular to devices offering more than just TCS- related services.

3.3 OBTAIN ACCESS RIGHTS

Using the Obtain access rights procedure, a device can obtain the rights to use the telephony services provided by another device, part of a WUG.

3.3.1 Procedure description

A device requests access rights by sending an ACCESS RIGHTS REQUEST message and starting timer T401. Upon receipt of the ACCESS RIGHTS REQUEST message, the receiving device accepts the request for access rights by sending an ACCESS RIGHTS ACCEPT.

When the requesting device receives the ACCESS RIGHTS ACCEPT, it shall stop timer T401. Then, the access rights procedure has completed successfully.

If no response has been received before the expiration of timer T401, the requesting device shall consider the request for access rights to be denied.

If, upon receipt of the ACCESS RIGHTS REQUEST message, the receiving device is for some reason unable to accept the access rights, it shall reply with an ACCESS RIGHTS REJECT message. Upon receipt of an ACCESS RIGHTS REJECT message, the requesting device shall stop timer T401 and consider the request for access rights to be denied.

3.3.2 Message flow

The figure below provides the complete view on the messages exchanged during the Obtain access rights procedure.

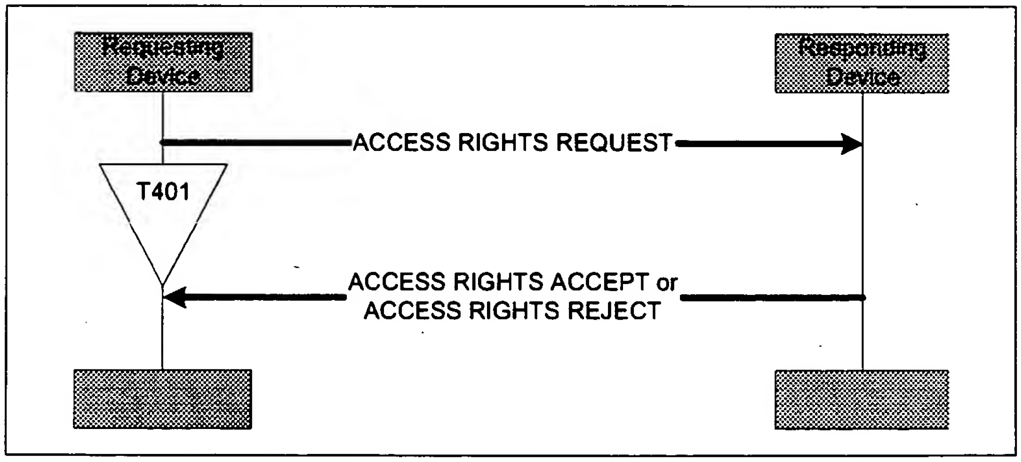

Figure 3.1: Obtain access rights message flow

3.4 CONFIGURATION DISTRIBUTION

The units in the WUG need to be informed about changes in the WUG; e.g. when a unit is added or removed. The Configuration distribution procedure is used to exchange this data.

When a WUG configuration change occurs, the WUG master initiates the Configuration distribution procedure on all WUG members. The WUG master keeps track of which WUG members have been informed of WUG configuration changes.

Some WUG members may be out of range and may therefore not be reached. The update of these WUG members will be performed when these members renew contact with the WUG master.

3.4.1 Procedure Description

The WUG master initiates the Configuration distribution procedure by starting timer T403, and transferring the INFO SUGGEST message. The INFO SUGGEST message contains the complete WUG configuration information. Upon receipt of the INFO SUGGEST message, the WUG member shall send an INFO ACCEPT message, to acknowledge the proper receipt of the WUG configuration information.

When the WUG master receives the INFO ACCEPT, the timer T401 is stopped, and the Configuration distribution procedure has completed successfully. On expiry of timer T403, the Configuration distribution procedure is terminated.

3.4.2 Message flow

The figure below provides the complete view on the messages exchanged during the Configuration distribution procedure, as described in the sections above.

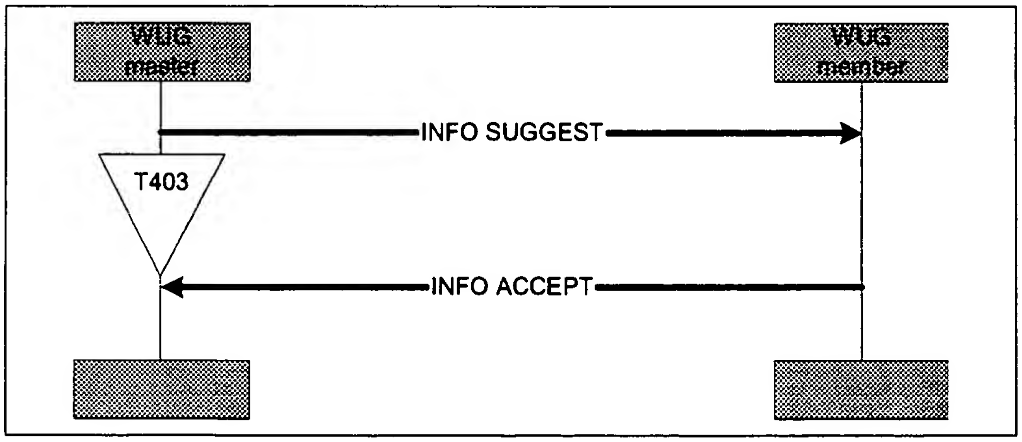

Figure 3.2: Configuration distribution message flow

3.5 FAST INTER-MEMBER ACCESS

When two WUG members are both active in the WUG master piconet, a WUG member can use the Fast inter-member access procedure to obtain fast access to another WUG member. With the Fast inter-member access procedure, the originating WUG member obtains clock information from the terminating WUG member and forces the terminating WUG member to go into PAGE_SCAN for a defined period (T406).

3.5.1 Listen Request

The originating WUG member initiates the Fast inter-member access procedure by starting timer T404 and transferring the LISTEN REQUEST message to the WUG master, indicating the WUG member with which it wishes to establish contact.

If, before expiry of timer T404, the originating WUG member receives no response to the LISTEN REQUEST message, the originating WUG member shall terminate the procedure.

3.5.2 Listen Accept

Upon receipt of the LISTEN REQUEST message, the WUG master checks that the indicated WUG member is part of the WUG. If this is the case, the WUG master initiates the Fast inter-member access towards the terminating WUG member side by starting timer T405 and sending the LISTEN SUGGEST message to the terminating WUG member.

Upon receipt of the LISTEN SUGGEST message, the terminating WUG member confirms the suggested action (internal call) by sending a LISTEN ACCEPT message to the WUG master. This message contains the terminating WUG member's clock offset. After sending the LISTEN ACCEPT, the terminating WUG member shall go to PAGE-SCAN state, for T406 seconds, to enable connection establishment by the originating WUG member.

Upon receipt of the LISTEN ACCEPT message, the WUG master stops timer T405, and informs the originating WUG member of the result of the WUG fast inter-member access by sending a LISTEN ACCEPT message. This message contains the terminating WUG member's clock offset. Upon receipt of the LISTEN ACCEPT message, the originating WUG member stops timer T404, and starts paging the terminating WUG member.

If no response to the LISTEN SUGGEST message is received by the WUG master before the first expiry of timer T405, then the WUG master shall terminate the Fast inter-member access procedure by sending a LISTEN REJECT message to both originating and terminating WUG member using cause #102, *recovery on timer expiry.*

3.5.3 Listen Reject by the WUG Master

If the WUG master rejects the Fast inter-member access procedure, it sends a LISTEN REJECT message to the originating WUG member.

Valid cause values are:
#1, *Unallocated (unassigned) number* (when the indicated WUG member is not part of the WUG)
#17, *User busy* (in case terminating WUG member is engaged in an external call)
#20, *Subscriber absent* (upon failure to establish contact with the terminating WUG member), or
any cause value indicated in a LISTEN REJECT message received from/sent to the terminating WUG member.

Upon receipt of the LISTEN REJECT message, the originating WUG member stops timer T404, and terminates the procedure.

3.5.4 Listen Reject by the WUG Member

If the terminating WUG member rejects the suggested action received in the LISTEN SUGGEST message, it sends a LISTEN REJECT message to the WUG master. Valid cause value is #17, *User busy* (in case terminating WUG member is engaged in another internal call).

Upon receipt of the LISTEN REJECT, the WUG master stops timer T405, and continues as described in Section 3.5.3 on page 454.

3.5.5 Message flow

The figure below provides a view of the messages exchanged during Fast inter-member access, as described in the sections above. A successful Fast inter-member access procedure ends with the terminating WUG member going into page scan, thus allowing the originating WUG member to contact him directly.

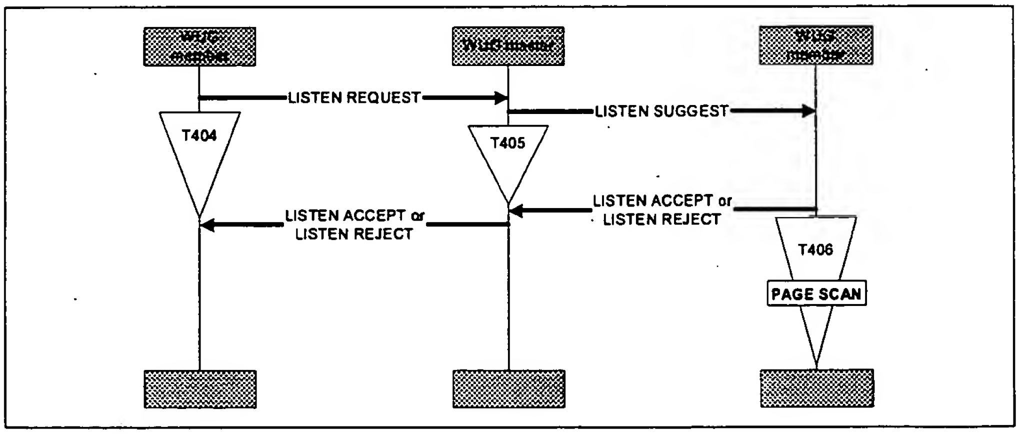

Figure 3.3: Fast inter-member access message flow

4 CONNECTIONLESS TCS (CL)

A connectionless TCS message can be used to exchange signalling information without establishing a TCS call. It is thus a connectionless service offered by TCS.

A connectionless TCS message is a CL INFO message (as defined in Section 6.3.1 on page 470).

A connection-oriented L2CAP channel between the Outgoing and Incoming Side shall be available before a CL INFO message can be sent.

Note: In the case of a connection-oriented channel, it may choose to delay the termination of the channel for a defined period to exchange more CL INFO messages.

Alternatively, in a multi-point configuration (see Section 1.2 on page 435), a connectionless L2CAP channel may be used and, if so, shall be available before a CL INFO can be sent.

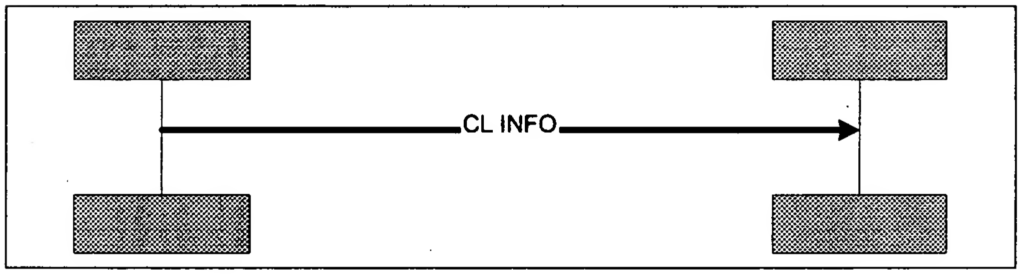

Figure 4.1: Connectionless TCS message flow

5 SUPPLEMENTARY SERVICES (SS)

The TCS provides explicit support for only one supplementary service, the Calling Line Identity (seeSection 5.1 on page 456).

For supplementary services provided by an external network, using DTMF sequences for the activation/de-activation and interrogation of supplementary services, the DTMF start & stop procedure is supported (see Section 5.2 on page 456). This procedure allows both finite and infinite tone lengths.

Section 5.3 on page 458 specifies how a specific supplementary service, provided by an external network, called register recall is supported.

For other means of supplementary service control, no explicit support is specified. Support may be realized by either using the service call, or use the company specific information element, or a combination.

5.1 CALLING LINE IDENTITY

To inform the Incoming Side of the identity of the originator of the call, the Outgoing Side may include the calling party number information element (see Section 7.4.6 on page 481) in the SETUP message transferred as part of the call request.

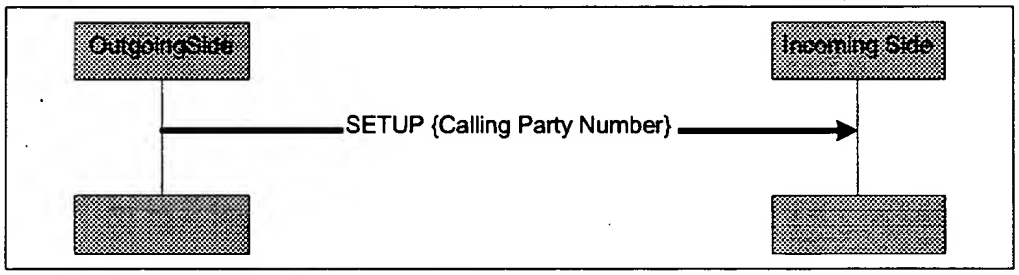

Figure 5.1: Calling line identity message flow

5.2 DTMF START & STOP

The DTMF start & stop procedure is supported to provide supplementary service control on PSTN type of networks.

In principle DTMF messages can be initiated by either (Outgoing or Incoming) Side; in practice, however, the Side (gateway) connected to the external PSTN network will be the recipient.

DTMF messages can be transmitted only in the active state of a call. Tone generation shall end when the call is disconnected.

5.2.1 Start DTMF request

A user may cause a DTMF tone to be generated; e.g. by depression of a key. The relevant action is interpreted as a requirement for a DTMF digit to be sent in a START DTMF message on an established signalling channel. This message contains the value of the digit to be transmitted (0, 1...9, A, B, C, D, *, #).

Only a single digit will be transferred in each START DTMF message.

5.2.2 Start DTMF response

The side receiving the START DTMF message will reconvert the received digit back into a DTMF tone which is applied toward the remote user, and return a START DTMF ACKNOWLEDGE message to the initiating side. This acknowledgment may be used to generate an indication as a feedback for a successful transmission.

If the receiving side cannot accept the START DTMF message, a START DTMF REJECT message will be sent to the initiating side, using cause #29, *Facility rejected*, indicating that sending DTMF is not supported by the external network.

5.2.3 Stop DTMF request

When the user indicates the DTMF sending should cease (e.g. by releasing the key) the initiating side will send a STOP DTMF message to the other side.

5.2.4 Stop DTMF response

Upon receiving the STOP DTMF message, the receiving side will stop sending the DTMF tone (if still being sent) and return a STOP DTMF ACKNOWLEDGE message to the initiating side.

5.2.5 Message flow

The figure below provides a view of the messages exchanged when a single DTMF tone needs to be generated.

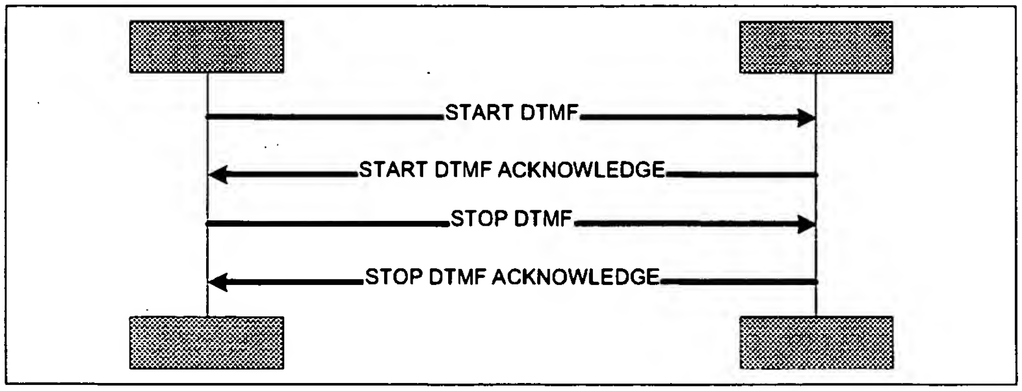

Figure 5.2: DTMF start & stop message flow

5.3 REGISTER RECALL

Register recall means to seize a register (with dial tone) to permit input of further digits or other action. In some markets, this is referred to as 'hook flash'. Register recall is supported by sending an INFORMATION message with a keypad facility information element, indicating 'register recall' (value 16H). Further digits are sent using the procedures as indicated in Section 5.2 above.

6 MESSAGE FORMATS

This section provides an overview of the structure of messages used in this specification, and defines the function and information contents (i.e. semantics) of each message.

Whenever a message is sent according to the procedures of Sections 2, 3 and 4, it shall contain the mandatory information elements, and optionally any combination of the optional information elements specified in this section for that message.

A message shall always be delivered in a single L2CAP packet. The start of a message (the LSB of the first octet) shall be aligned with the start of the L2CAP payload.

Each definition includes:

a) A brief description of the message direction and use

b) A table listing the information elements in order of their appearance in the message (same relative order for all message types)

c) Indications for each information element in the table, specifying –

- the section of this specification describing the information element
- whether inclusion in mandatory ('M') or optional ('O')
- the length (or length range) of the information element, where '*' denotes an undefined maximum length which may be application dependent.

d) Further explanatory notes, as necessary

All message formats are denoted in octets.

6.1 CALL CONTROL MESSAGE FORMATS

6.1.1 ALERTING

This message is sent by the incoming side to indicate that the called user alerting has been initiated.

Message Type: ALERTING

Direction: incoming to outgoing

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Bearer capability	7.4.3	O Note 1)	4(26)
Progress indicator	7.4.13	O	2
SCO Handle	7.4.14	O	2
Destination CID	7.4.11	O	4
Company specific	7.4.9	O	3-*

Table 6.1: ALERTING message content

Note 1: Allowed only in the first message sent by the incoming side.

6.1.2 CALL PROCEEDING

This message is sent by the incoming side to indicate that the requested call establishment has been initiated and no more call establishment information will be accepted.

Message Type: CALL PROCEEDING

Direction: incoming to outgoing

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Bearer capability	7.4.3	O Note 1)	4(26)
Progress indicator	7.4.13	O	2
SCO Handle	7.4.14	O	2
Destination CID	7.4.11	O	4
Company specific	7.4.9	O	3-*

Table 6.2: CALL PROCEEDING message content

Note 1: Allowed only in the first message sent by the incoming side.

6.1.3 CONNECT

This message is sent by the incoming side to indicate call acceptance by the called user.

Message Type: CONNECT

Direction: incoming to outgoing

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Bearer capability	7.4.3	O^Note 1)	4(26)
SCO Handle	7.4.14	O	2
Company specific	7.4.9	O	3-*

Table 6.3: CONNECT message content

Note 1: Allowed only in the first message sent by the incoming side.

6.1.4 CONNECT ACKNOWLEDGE

This message is sent by the outgoing side to acknowledge the receipt of a CONNECT message.

Message Type: CONNECT ACKNOWLEDGE

Direction: outgoing to incoming

Information Element	Ref.	Type	Length
Message type	7.3	M	1
SCO Handle	7.4.14	O	2
Destination CID	7.4.11	O	4
Company specific	7.4.9	O	3-*

Table 6.4: CONNECT ACKNOWLEDGE message content

6.1.5 DISCONNECT

This message is sent by either side as an invitation to terminate the call.

Message Type: DISCONNECT

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Cause	7.4.7	O	2
Progress indicator	7.4.13	O	2
SCO Handle	7.4.14	O	2
Destination CID	7.4.11	O	4
Company specific	7.4.9	O	3-*

Table 6.5: DISCONNECT message content

6.1.6 INFORMATION

This message is sent by either side to provide additional information during call establishment (in case of overlap sending).

Message Type: INFORMATION

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Sending complete	7.4.15	O	1
Keypad facility	7.4.12	O	2
Called party number	7.4.5	O	3-*
Audio control	7.4.2	O	3-*
Company specific	7.4.9	O	3-*

Table 6.6: INFORMATION message content

6.1.7 PROGRESS

This message is sent by the incoming side to indicate the progress of a call in the event of interworking or by either side in the call with the provision of optional in-band information/patterns.

Message Type: PROGRESS

Direction: incoming to outgoing

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Progress indicator	7.4.13	M	2
SCO Handle	7.4.14	O	2
Destination CID	7.4.11	O	4
Company specific	7.4.9	O	3-*

Table 6.7: PROGRESS message content

6.1.8 RELEASE

This message is used to indicate that the device sending the message had disconnected the channel (if any) and intends to release the channel, and that receiving device should release the channel after sending RELEASE COMPLETE.

Message Type: RELEASE

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Cause	7.4.7	O^Note 1)	2
Company specific	7.4.9	O	3-*

Table 6.8: RELEASE message content

Note 1: Mandatory in the first call clearing message.

6.1.9 RELEASE COMPLETE

This message is used to indicate that the device sending the message has released the channel (if any), and that the channel is available for re-use.

Message Type: RELEASE COMPLETE

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Cause	7.4.7	O^Note 1)	2
Company specific	7.4.9	O	3-*

Table 6.9: RELEASE COMPLETE message content

Note 1: Mandatory in the first call clearing message.

6.1.10 SETUP

This message is sent by the outgoing side to initiate call establishment.

Message Type:

Direction:

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Call class	7.4.4	M	2
Sending complete	7.4.15	O	1
Bearer capability	7.4.3	O	4(26)
Signal	7.4.16	O	2
Calling party number	7.4.6	O	3-*
Called party number	7.4.5	O	3-*
Company specific	7.4.9	O	3-*

Table 6.10: SETUP message content

6.1.11 SETUP ACKNOWLEDGE

This message is sent by the incoming side to indicate that call establishment has been initiated, but additional information may be required.

Message Type: SETUP ACKNOWLEDGE

Direction: incoming to outgoing

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Bearer capability	7.4.3	O[Note 1)]	4(26)
Progress indicator	7.4.13	O	2
SCO Handle	7.4.14	O	2
Destination CID	7.4.11	O	4
Company specific	7.4.9	O	3-*

Table 6.11: SETUP ACKNOWLEDGE message content

Note 1: Allowed only in the first message sent by the incoming side.

6.1.12 Start DTMF

This message contains the digit the other side should reconvert back into a DTMF tone, which is then applied towards the remote user.

Message Type: Start DTMF

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Keypad facility	7.4.12	M	2

Table 6.12: Start DTMF message content

6.1.13 Start DTMF Acknowledge

This message is sent to indicate the successful initiation of the action required by the Start DTMF message.

Message Type: Start DTMF Acknowledge

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Keypad facility	7.4.12	M	2

Table 6.13: Start DTMF Acknowledge message content

6.1.14 Start DTMF Reject

This message is sent to indicate that the other side cannot accept the Start DTMF message.

Message Type: Start DTMF Reject

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Cause	7.4.7	O	2

Table 6.14: Start DTMF Reject message content

6.1.15 Stop DTMF

This message is used to stop the DTMF tone sent towards the remote user.

Message Type: Stop DTMF

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1

Table 6.15: Stop DTMF message content

6.1.16 Stop DTMF Acknowledge

This message is sent to indicate that the sending of the DTMF tone has been stopped.

Message Type: Stop DTMF Acknowledge

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Keypad facility	7.4.12	M	2

Table 6.16: Stop DTMF Acknowledge message content

6.2 GROUP MANAGEMENT MESSAGE FORMATS

6.2.1 ACCESS RIGHTS REQUEST

This message is sent by the initiating side to obtain access rights.

Message Type: ACCESS RIGHTS REQUEST

Direction:

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Company specific	7.4.9	O	3-*

Table 6.17: ACCESS RIGHTS REQUEST message content

6.2.2 ACCESS RIGHTS ACCEPT

This message is sent by the responding side to indicate granting of access rights.

Message Type: ACCESS RIGHTS ACCEPT

Direction:

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Company specific	7.4.9	O	3-*

Table 6.18: ACCESS RIGHTS ACCEPT message content

6.2.3 ACCESS RIGHTS REJECT

This message is sent by the responding side to indicate denial of access rights.

Message Type: ACCESS RIGHTS REJECT

Direction:

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Company specific	7.4.9	O	3-*

Table 6.19: ACCESS RIGHTS REJECT message content

6.2.4 INFO SUGGEST

This message is sent by the WUG master to indicate that a change has occurred in the WUG configuration.

Message Type: INFO SUGGEST

Direction: WUG master to WUG member

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Configuration Data	7.4.10	M	*
Company specific	7.4.9	O	3-*

Table 6.20: INFO SUGGEST message content

6.2.5 INFO ACCEPT

This message is sent by the WUG member to indicate the acceptance of the updated WUG configuration.

Message Type: INFO ACCEPT

Direction: WUG member to WUG master

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Company specific	7.4.9	O	3-*

Table 6.21: INFO ACCEPT message content

6.2.6 LISTEN REQUEST

This message is sent by a WUG member to indicate to the WUG master the request for a Fast inter-member access to the indicated WUG member.

Message Type: LISTEN REQUEST

Direction: WUG member to WUG master

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Called party number	7.4.6	M	3-*
Company specific	7.4.9	O	3-*

Table 6.22: LISTEN REQUEST message content

6.2.7 LISTEN SUGGEST

This message is sent by a WUG master to indicate to the WUG member the request for a Fast inter-member access.

Message Type: LISTEN SUGGEST

Direction: WUG master to WUG member

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Company specific	7.4.9	O	3-*

Table 6.23: LISTEN SUGGEST message content

6.2.8 LISTEN ACCEPT

This message is sent to indicate the acceptance of the previous request for a Fast inter-member access.

Message Type: LISTEN ACCEPT

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Clock offset	7.4.8	O	4
Company specific	7.4.9	O	3-*

Table 6.24: LISTEN ACCEPT message content

6.2.9 LISTEN REJECT

This message is sent to indicate the rejection of the previous request for a Fast inter-member access.

Message Type: LISTEN REJECT

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Cause	7.4.7	O	2
Company specific	7.4.9	O	3-*

Table 6.25: LISTEN REJECT message content

6.3 TCS CONNECTIONLESS MESSAGE FORMATS

6.3.1 CL INFO

This message is sent by either side to provide additional information in a connectionless manner.

Message Type: CL INFO

Direction: both

Information Element	Ref.	Type	Length
Message type	7.3	M	1
Audio control	7.4.2	O	3-*
Company specific	7.4.9	O	3-*

Table 6.26: CL INFO message content

7 MESSAGE CODING

The figures and text in this section describe message contents. Within each octet, the bit designated 'bit 1' is transmitted first, followed by bit 2, 3, 4, etc. Similarly, the octet shown at the top of the figure is sent first.

Whenever a message is sent, according to the procedures of Sections 2, 3 and 4, it shall be coded as specified in this section.

7.1 OVERVIEW

The coding rules follow ITU-T Recommendation Q.931, but is tailored to the specific needs of TCS.

Every message consists of:

a) Protocol discriminator

b) Message type, and

c) Other information elements, as required

The Protocol discriminator and Message type is part of every TCS message, while the other information elements are specific to each message type.

8 7 6	5 4 3 2 1	
Protocol discriminator	Message type	octet 1
Other information elements as required		octet 2

Table 7.1: General message format

A particular information element shall be present only once in a given message.

The term 'default' implies that the value defined shall be used in the absence of any assignment or negotiation of alternative values.

For notation purposes – when a field extends over more than one octet, the order of bit values progressively decreases as the octet number increases. The least significant bit of the field is represented by the lowest numbered bit of the highest-numbered octet of the field. In general, bit 1 of each octet contains the least significant bit of a field.

7.2 PROTOCOL DISCRIMINATOR

The purpose of the protocol discriminator is to distinguish the TCS messages into different functional groups. The protocol discriminator is the first part of every message.

The protocol discriminator is coded according to Figure 7.1 and Table 7.2.

8	7	6	5	4	3	2	1	
Protocol discriminator			▨	▨	▨	▨	▨	octet 1

Figure 7.1: Protocol discriminator

Bits			
8	**7**	**6**	
0	0	0	Bluetooth TCS Call Control
0	0	1	Bluetooth TCS Group management
0	1	0	Bluetooth TCS Connectionless
All other values reserved			

Table 7.2: Protocol discriminator

7.3 MESSAGE TYPE

The purpose of the message type is to identify the function of the message being sent.

The Message type is the first part of every message and it is coded as shown in Figure 7.2 and Table 7.3.

8	7	6	5	4	3	2	1	
▨	▨	▨	Message type					octet 1

Figure 7.2: Message type

Bits					
5	**4**	**3**	**2**	**1**	
					Call Control messages
					Call Establishment
0	0	0	0	0	ALERTING

Table 7.3: Message type

Bits					
5	4	3	2	1	
0	0	0	0	1	CALL PROCEEDING
0	0	0	1	0	CONNECT
0	0	0	1	1	CONNECT ACKNOWLEDGE
0	0	1	0	0	PROGRESS
0	0	1	0	1	SETUP
0	0	1	1	0	SETUP ACKNOWLEDGE
					Call clearing
0	0	1	1	1	DISCONNECT
0	1	0	0	0	RELEASE
0	1	0	0	1	RELEASE COMPLETE
					Miscellaneous
0	1	0	1	0	INFORMATION
1	0	0	0	0	START DTMF
1	0	0	0	1	START DTMF ACKNOWLEDGE
1	0	0	1	0	START DTMF REJECT
1	0	0	1	1	STOP DTMF
1	0	1	0	0	STOP DTMF ACKNOWLEDGE
					Group management messages
0	0	0	0	0	INFO SUGGEST
0	0	0	0	1	INFO ACCEPT
0	0	0	1	0	LISTEN REQUEST
0	0	0	1	1	LISTEN ACCEPT
0	0	1	0	0	LISTEN SUGGEST
0	0	1	0	1	LISTEN REJECT
0	0	1	1	0	ACCESS RIGHTS REQUEST
0	0	1	1	1	ACCESS RIGHTS ACCEPT
0	1	0	0	0	ACCESS RIGHTS REJECT
					Connectionless messages
0	0	0	0	0	CL INFO

Table 7.3: Message type

7.4 OTHER INFORMATION ELEMENTS

7.4.1 Coding rules

The coding of other information elements follows the coding rules described below.

Three categories of information elements are defined:

a) single octet information elements (see Figure 7.3 on page 474)

b) double octet information element (see Figure 7.4 on page 474)

c) variable length information elements (see Figure 7.5 on page 474).

Table 7.4 on page 474 summarizes the coding of the information element identified bits for those information elements used in this specification.

8	7	6	5	4	3	2	1	
1	Information element identifier							octet 1

Figure 7.3: Single octet information element format

8	7	6	5	4	3	2	1	
1	Information element identifier							octet 1
Contents of information element								octet 2

Figure 7.4: Double octet information element format

8	7	6	5	4	3	2	1	
0	Information element identifier							octet 1
Length of contents of information element (octets)								octet 2
Contents of Information element								octet 3, etc.

Figure 7.5: Variable length information element format

Coding									Ref.	Max Length (octets)
8	7	6	5	4	3	2	1			
1								*Single octet information elements*		
	0	1	0	0	0	0	1	Sending complete	7.4.15	1
1								*Double octet information elements*		

Table 7.4: Information element identifier coding

Coding									Ref.	Max Length (octets)
8	7	6	5	4	3	2	1			
	1	0	0	0	0	0	0	Call class	7.4.4	2
	1	0	0	0	0	0	1	Cause	7.4.7	2
	1	0	0	0	0	1	0	Progress indicator	7.4.13	2
	1	0	0	0	0	1	1	Signal	7.4.16	2
	1	0	0	0	1	0	0	Keypad facility	7.4.12	2
	1	0	0	0	1	0	1	SCO handle	7.4.14	2
0								*Variable length information elements*		
	0	0	0	0	0	0	0	Clock offset	7.4.8	4
	0	0	0	0	0	0	1	Configuration data	7.4.2	*
	0	0	0	0	0	1	0	Bearer capability	7.4.3	4(26)
	0	0	0	0	0	1	1	Destination CID	7.4.11	4
	0	0	0	0	1	0	0	Calling party number	7.4.6	*
	0	0	0	0	1	0	1	Called party number	7.4.5	*
	0	0	0	0	1	1	0	Audio control	7.4.2	*
	0	0	0	0	1	1	1	Company specific	7.4.9	*

Table 7.4: Information element identifier coding

The descriptions of the information elements below are organized in alphabetical order. However, there is a particular order of appearance for each information element in a message. The code values of the information element identifier for the variable length formats are assigned in ascending numerical order, according to the actual order of appearance of each information element in a message. This allows the receiving devices to detect the presence or absence of a particular information element without scanning through an entire message.

Where the description of information elements in this specification contains spare bits, these bits are indicated as being set to '0'. In order to allow compatibility with future implementation, messages should not be rejected simply because a spare bit is set to '1'.

The second octet of a variable length information element indicates the total length of the contents of that information element regardless of the coding of the first octet (i.e. the length is calculated starting from octet 3). It is the binary coding of the number of octets of the contents, with bit 1 as the least significant bit (2^0).

An optional variable-length information element may be present, but empty (zero length). The receiver should interpret this as if that information element was absent. Similarly, an absent information element should be interpreted by the receiver as if that information element was empty.

7.4.2 Audio control

The purpose of the Audio control information elements is to indicate information relating to the control of audio.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	1	0	1
Length of contents of information element (octets)								2
Control information								3

Figure 7.6:

Control information (octet 3)							
Bits							
7	6	5	4	3	2	1	
0	0	0	0	0	0	0	Volume increase
0	0	0	0	0	0	1	Volume decrease
0	0	0	0	0	1	0	Microphone gain increase
0	0	0	0	0	1	1	Microphone gain decrease
0	X	X	X	X	X	X	Reserved for Bluetooth standardization
1	X	X	X	X	X	X	Company specific

Table 7.5: Audio Control information element coding

7.4.3 Bearer capability

The purpose of the Bearer capability information elements is to indicate a requested or available bearer service.

If this information element is absent, the default Bearer capability is Link type Synchronous Connection-Oriented with packet type HV3, using CVSD coding for the User information layer 1.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	1	0	1
Length of contents of information element (octets)								2
Link type								3

Figure 7.7:

Link type element coding = 00000000 (SCO)

User information layer 1	Packet type	4

Figure 7.8:

Link type element coding = 00000001 (ACL)

Flags		4
Service type		5
Token Rate		6
		7
		8
		9
Token Bucket Size (bytes)		10
		11
		12
		13
Peak Bandwidth (bytes/second)		14
		15
		16
		17
Latency (microseconds)		18
		19
		20
		21
Delay Variation (microseconds)		22
		23
		24
		25
User information layer 3	User information layer 2	26

Figure 7.9:

Note: the Quality of Service is repeated at TCS level, as only TCS has the knowledge of end-to-end Quality of Service requirements.

Link type (octet 3)	
Bits	
8 7 6 5 4 3 2 1	
0 0 0 0 0 0 0 0	Synchronous Connection-Oriented
0 0 0 0 0 0 0 1	Asynchronous Connection-Less
0 0 0 0 0 0 1 0	None
All other values are reserved	
Octet 4 coding (Link type element coding = 000000000)	
Packet type (octet 4)	
Bits	
5 4 3 2 1	
0 0 1 0 1	HV1
0 0 1 1 0	HV2
0 0 1 1 1	HV3
0 1 0 0 0	DV
All other values are reserved	
User information layer 1 (octet 4)	
Bits	
8 7 6	
0 0 1	CVSD
0 1 0	PCM A-law
0 1 1	PCM µ-law
All other values reserved	
Octets 4-26 coding (Link type element coding = 000000001)	
The details of the coding Octets 4-25 can be found in L2CAP, see L2CAP, Section 6 on page 289	
User information layer 2 (octet 26)	
Bits	
4 3 2 1	
0 0 0 0	RFCOMM over L2CAP
All other values are reserved	
User information layer 3 (octet 26)	
Bits	
8 7 6 5	
0 0 0 0	Not specified
0 0 0 1	PPP
0 0 1 0	IP
All other values reserved	
Octet 4 coding (Link type element coding = 000000010)	
Octet 4 is absent	

Table 7.6: Bearer capability information element coding

7.4.4 Call class

The purpose of the Call class is to indicate the basic aspects of the service requested. This element allows the user to indicate the use of default attributes, thereby reducing the length of the set-up message.

8	7	6	5	4	3	2	1	Octets
1	1	0	0	0	0	0	0	1
Call Class								2

Figure 7.10:

Call class (octet 2)		
	Bits	
8	7 6 5 4 3 2 1	
0	0 0 0 0 0 0 0	External call
0	0 0 0 0 0 0 1	Intercom call
0	0 0 0 0 0 1 0	Service call
0	0 0 0 0 0 1 1	Emergency call
	All other values reserved	

Table 7.7: Call class information element coding

Note

- An external call is a call to/from an external network; e.g. the PSTN.
- An intercom call is a call between Bluetooth devices.
- A service call is a call for configuration purposes.
- An emergency call is an external call using a dedicated emergency call number, using specific properties.

7.4.5 Called party number

The purpose of the Called party number information element is to identify the called party of a call.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	0	1	1
Length of contents of information element (octets)								2
0	Type of number			Numbering plan identification				3
0	Number digits (IA5 characters) (Note)							4 etc.

Note – The number digits appear in multiple octet 4's in the same order in which they would be entered, that is, the number digit which would be entered first is located in the first octet 4.

Figure 7.11:

Type of number (octet 3)

Bits 7 6 5	
0 0 0	Unknown
0 0 1	International number
0 1 0	National number
0 1 1	Network specific number
1 0 0	Subscriber number
1 1 0	Abbreviated number
1 1 1	Reserved for extension
All other values are reserved	

Numbering plan identification (octet 3)

Bits 4 3 2 1	
0 0 0 0	Unknown
0 0 0 1	ISDN/telephony numbering plan E.164
0 0 1 1	Data numbering plan Rec. X.121
0 1 0 0	Reserved
1 0 0 0	National standard numbering plan
1 0 0 1	Private numbering plan
All other values are reserved	

Table 7.8: Called party information element coding

7.4.6 Calling party number

The purpose of the Calling party number information element is to identify the origin of a call.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	0	0	1
Length of contents of information element (octets)								2
0	Type of number			Numbering plan identification				3
0	Presentation indicator		0	0	0	Screening indicator		4
0	Number digits (IA5 characters)							5 etc.

Figure 7.12:

Type of number (octet 3)

Bits 7 6 5	
0 0 0	Unknown
0 0 1	International number
0 1 0	National number
0 1 1	Network specific number
1 0 0	Subscriber number
1 1 0	Abbreviated number
1 1 1	Reserved for extension
All other values are reserved	

Numbering plan identification (octet 3)

Bits 4 3 2 1	
0 0 0 0	Unknown
0 0 0 1	ISDN/telephony numbering plan E.164
0 0 1 1	Data numbering plan Rec. X.121
0 1 0 0	Reserved
1 0 0 0	National standard numbering plan
1 0 0 1	Private numbering plan
All other values are reserved	

Presentation indicator (octet 4)

Bits 7 6	
0 0	Presentation allowed
0 1	Presentation restricted
1 0	Number not available due to interworking
1 1	Reserved
All other values are reserved	

Table 7.9: Calling party information element coding

Screening indicator (octet 4)		
	Bits	
	2 1	
	0 0	User-provided, not screened
	0 1	User-provided, verified and passed
	1 0	User-provided, verified and failed
	1 1	Network provided
	All other values are reserved	

Table 7.9: Calling party information element coding

7.4.7 Cause

The purpose of the Cause is to indicate the remote side of the cause of the failure of the requested service.

8	7	6	5	4	3	2	1	Octets
1	1	0	0	0	0	0	1	1
Cause value								2

Figure 7.13:

Cause (octet 2)	
	Bits
8	7 6 5 4 3 2 1
0	These 7 bits are coded alike the Cause value subfield defined in Section 2.2.5 of ITU-T Recommendation Q.850[2].

Table 7.10: Cause information element coding

7.4.8 Clock offset

The purpose of the Clock offset information element is to indicate the Bluetooth clock offset used.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	0	0	1
Length of contents of information element (octets)								2
Clock offset								3
								4

Figure 7.14:

Clock offset coding (octet 3 and 4)															
	Bits (octet 3)							Bits (octet 4)							
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
0	Contains bits 16-2 of Bluetooth clock														

Table 7.11: Clock offset information element coding

7.4.9 Company specific

The purpose of the Company specific information element is to send non-standardized information.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	1	1	1
Length of contents of information element (octets)								2
Company Identification								3
Company Identification								4
Company specific contents								
								L+2

Figure 7.15:

Company identification coding (octet 3 and octet 4)																
	Bits (octet 3)							Bits (octet 4)								
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Ericsson
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Nokia Mobile Phones
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Intel Corporation
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	IBM Corporation
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Toshiba Corporation
	All other values are reserved															

Table 7.12: Company specific information element coding

7.4.10 Configuration data

The purpose of the Configuration data information element is to indicate the Configuration data.

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	0	1	1
Length of contents of information element (octets)								2
0	Internal number of WUG member 1 (IA5 characters)							3
0	Internal number of WUG member 1 (IA5 characters)							4
Bluetooth address of WUG member 1								5
								...
								10
Link key to be used towards WUG member 1								11
								..
								26
....								
0	Internal number of WUG member n (IA5 character)							**3+((n-1)*24)**
0	Internal number of WUG member n (IA5 character)							**4+((n-1)*24)**
Bluetooth address of WUG member n								**5+((n-1)*24)**
								...
								10+((n-1)*24)
Link key to be used towards WUG member n								**11+((n-1)*24)**
								..
								2+(n*24)

Note – The internal number (2 digits) appears in octets 3 and 4 in the same order in which they would be entered; that is, the number digit which would be entered first is located in octet 3.

Note – The octets 3-26 are repeated for all *n* WUG members.

Figure 7.16:

7.4.11 Destination CID

The purpose of the Destination CID information element is to enable the remote side to associate the established L2CAP channel with the ongoing call. The Destination CID is identical to the Destination CID (DCID) exchanged in the Configuration Request packet (see L2CAP, Section 5.4 on page 280).

8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	1	1	1
Length of contents of information element (octets)								2
DCID byte 1								3
DCID byte 0								4

Figure 7.17:

7.4.12 Keypad facility

The purpose of the Keypad facility information element is to convey IA5 characters; e.g. entered by means of a terminal keypad.

8	7	6	5	4	3	2	1	Octets
1	1	0	0	0	1	0	0	1
0	Keypad facility information (IA5 character)							2

Figure 7.18:

7.4.13 Progress indicator

The purpose of the Progress indicator information element is to describe an event that has occurred during the life of a call.

8	7	6	5	4	3	2	1	Octets
1	1	0	0	0	0	1	0	1
0	Progress description							2

Figure 7.19:

Progress information (octet 2)	
Bits 7 6 5 4 3 2 1	
0 0 0 1 0 0 0	In-band information or appropriate pattern is now available
All other values reserved	

Table 7.13: Progress indicator information element coding

7.4.14 SCO Handle

The purpose of the SCO handle information element is to enable the remote side to associate the established SCO link with the ongoing call. The SCO handle is identical to the SCO handle exchanged in the LMP_SCO_link_req sent by the piconet master (see LMP, Section 3.21 on page 219).

8	7	6	5	4	3	2	1	Octets
1	1	0	0	0	1	0	1	1
SCO handle value								2

Figure 7.20:

7.4.15 Sending complete

The purpose of the Sending complete information element is to optionally indicate completion of called party number.

8	7	6	5	4	3	2	1	Octet
1	0	1	0	0	0	0	1	1

Figure 7.21:

7.4.16 Signal

The purpose of the Signal information element is to convey information to a user regarding tones and alerting signals.

8	7	6	5	4	3	2	1	Octets
1	1	0	0	0	0	1	1	1
Signal value								2

Figure 7.22:

Signal value (octet 2)								
	Bits							
8	7	6	5	4	3	2	1	
0	1	0	0	0	0	0	0	External call
0	1	0	0	0	0	0	1	Internal call
0	1	0	0	0	0	1	0	Call back
0	X	X	X	X	X	X	X	Reserved for Bluetooth standardization
1	X	X	X	X	X	X	X	Company specific

Table 7.14: Signal information element coding

8 MESSAGE ERROR HANDLING[1]

8.1 PROTOCOL DISCRIMINATION ERROR

When a message is received with a protocol discriminator coded other than the ones defined in Section 7.2 on page 472, that message shall be ignored.

8.2 MESSAGE TOO SHORT OR UNRECOGNIZED

When a message is received that is too short to contain a complete message type information element, that message shall be ignored.

When a message is received that contains a complete message type information element, but with a value which is not recognized as a defined message type, that message shall be ignored.

8.3 MESSAGE TYPE OR MESSAGE SEQUENCE ERRORS

Whenever an unexpected message, except RELEASE or RELEASE COMPLETE message is received in any state other than the Null state, that message shall be ignored.

When an unexpected RELEASE message is received, the receiving side shall disconnect and release the bearer channel if established, return a RELEASE COMPLETE message, stop all timers, and enter the Null state.

When an unexpected RELEASE COMPLETE message is received, the receiving side shall disconnect and release the bearer channel if established, stop all timers, and enter the Null state.

8.4 INFORMATION ELEMENT ERRORS

The information elements in a message shall appear (if present for information elements indicated as optional) in the exact order as indicated in Section 6.

When a message is received which misses a mandatory information element, or which contains a mandatory information element with invalid content, the message shall be ignored.

In case the error occurred with a mandatory information element in a SETUP message, a RELEASE COMPLETE message shall be returned, either with cause #96, *mandatory information element is missing*, or with cause #100, *invalid information element contents*.

1. In this section, when it is stated to ignore a certain message or part of a message (information element), this shall be interpreted as to do nothing – as if the (part of the) message had never been received.

When a message is received which has an unrecognized information element, or has an optional information element with an invalid content, or has a recognized information element not defined to be contained in that message, the receiving side shall ignore the information element.

Information elements with a length exceeding the maximum length (as given in Section 7 on page 471) shall be treated as an information element with invalid content.

9 PROTOCOL PARAMETERS

9.1 PROTOCOL TIMERS

Timer name	Value
T301	Minimum 3 minutes
T302	15 seconds
T303	20 seconds
T304	30 seconds
T305	30 seconds
T308	4 seconds
T310	30 –120 seconds
T313	4 seconds
T401	8 second
T402	8 seconds
T403	4 second
T404	2.5 seconds
T405	2 seconds
T406	20 seconds

Table 9.1: Timer values

10 REFERENCES

[1] Q.931, "Digital Subscriber Signalling System No. 1(DSS 1) – ISDN User-Network interface Layer 3 Specification for Basic Call Control", 03/93

[2] Q.850, "Digital Subscriber Signalling System No. 1 General – Usage of cause of location in the Digital Subscriber Signalling system No. 1 and the signalling system No. 7 ISDN User Part", 03/93

11 LIST OF FIGURES

12 LIST OF TABLES

APPENDIX 1 - TCS CALL STATES

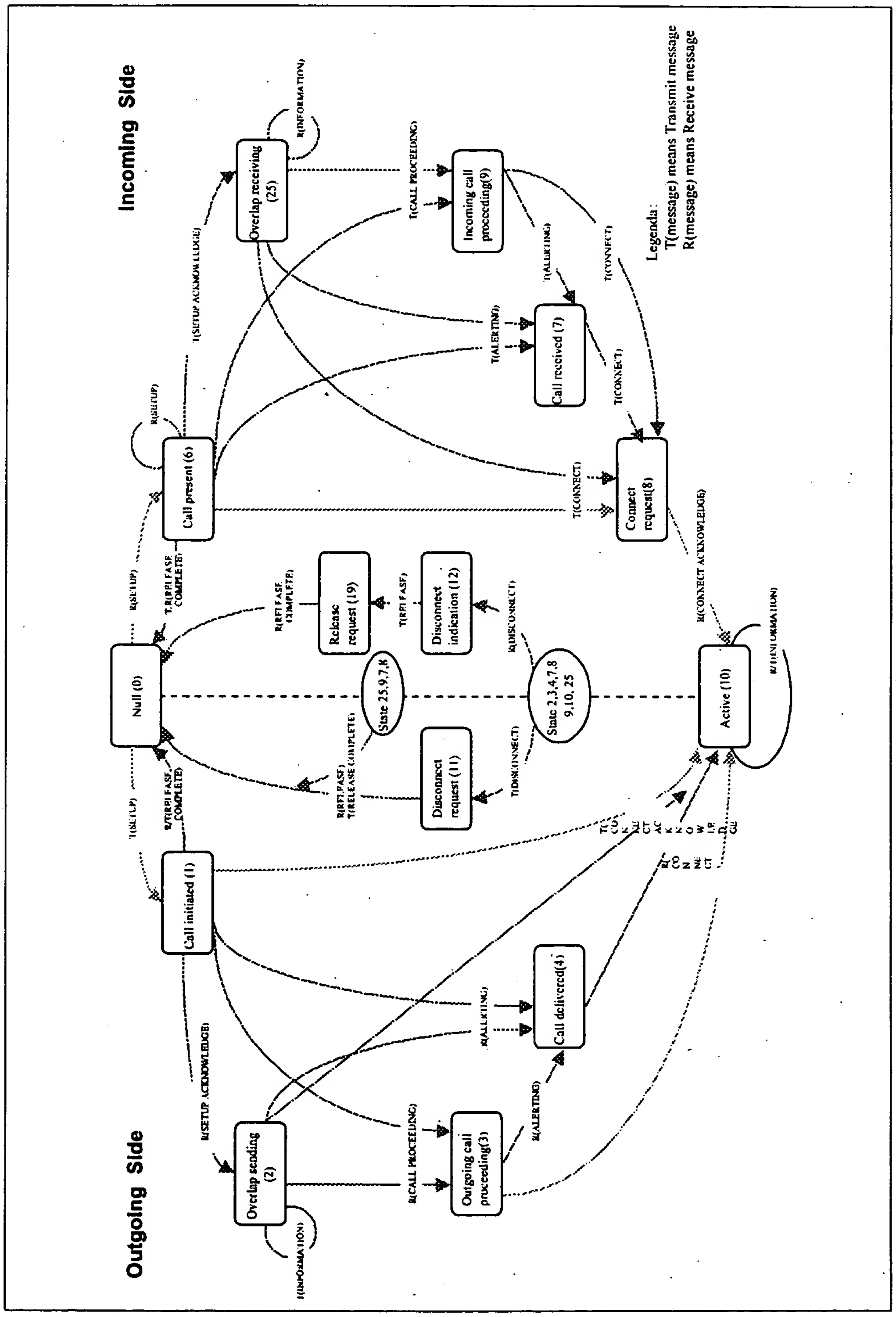

Figure A: Full TCS State Diagram

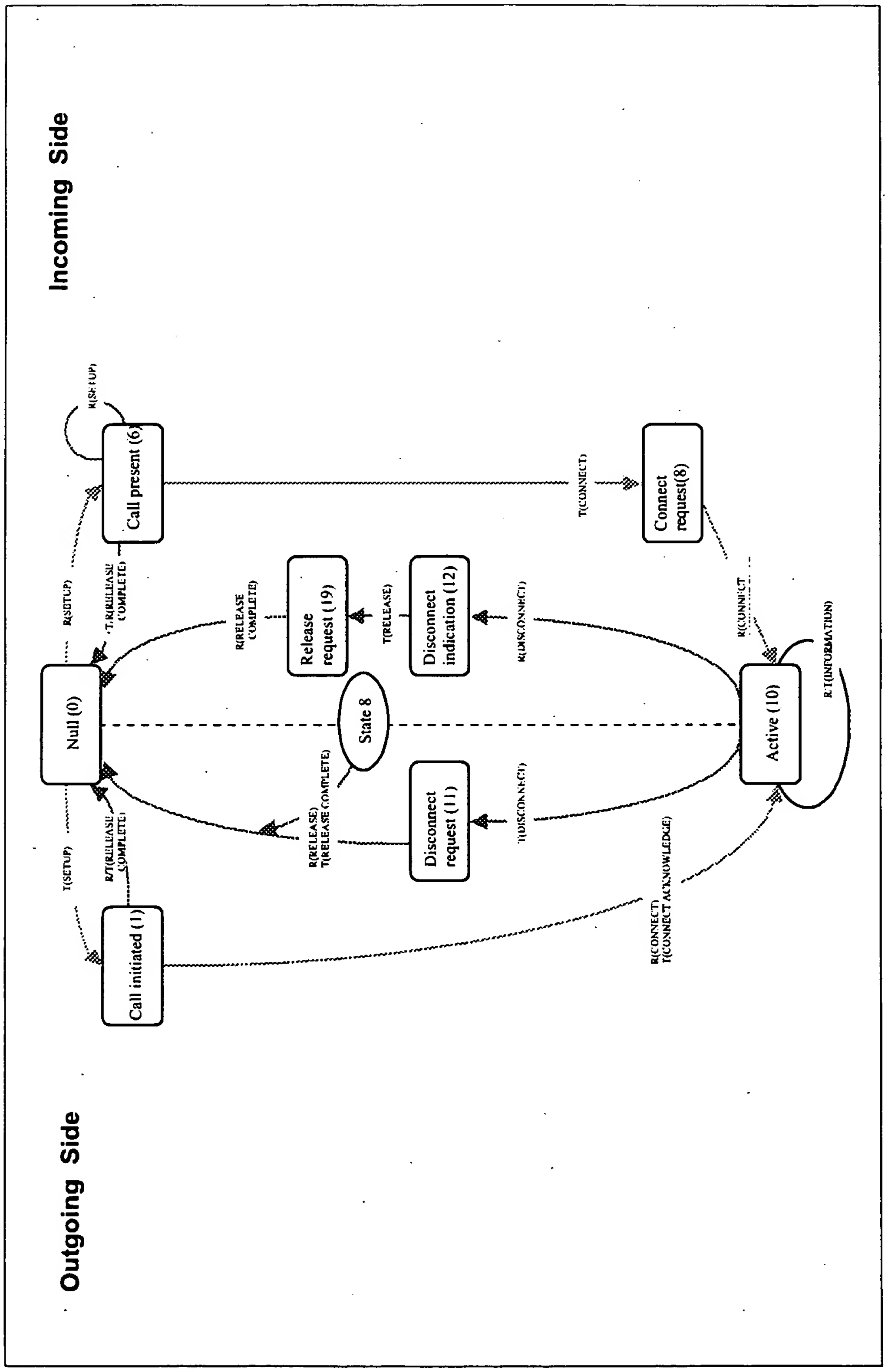

Figure B: Lean TCS State Diagram

Part F:4

INTEROPERABILITY REQUIREMENTS FOR BLUETOOTH AS A WAP BEARER

PPP Adaptation

Many of the characteristics of Bluetooth devices are shared with the target platforms for the Wireless Application Protocol. In some cases, the same device may be enabled for both types of communication. This document describes the interoperability requirements for using Bluetooth with PPP as the communications bearer for WAP protocols and applications.

CONTENTS

1 INTRODUCTION

1.1 DOCUMENT SCOPE

This document is intended for Bluetooth implementers who wish to take advantage of the dynamic, ad-hoc characteristics of the Bluetooth environment in providing access to value-added services using the WAP environment and protocols.

Bluetooth provides the physical medium and link control for communications between WAP client and server. This document describes how PPP may be used to achieve this communication.

The information contained in this document is not sufficient to allow the implementation of a general-purpose WAP client or server device. Instead, this document provides the following information:

- An overview of the use of WAP in the Bluetooth environment will explain how the concept of value-added services fits within the Bluetooth vision. Examples are given of how the WAP value-added services model can be used to fulfil specific Bluetooth usage models.
- The WAP Services Overview attempts to place the WAP environment in a familiar context. Each component of WAP is introduced, and is contrasted with equivalent Internet protocols (where applicable).
- A discussion of WAP in the Bluetooth Piconet describes how the particular structure of Bluetooth communications relates to WAP behaviors.
- Finally, the Interoperability Requirements describe the specific Bluetooth features that must be implemented in order to ensure interoperability between any two WAP enabled Bluetooth devices.

2 THE USE OF WAP IN THE BLUETOOTH ENVIRONMENT

2.1 VALUE-ADDED SERVICES

The presence of communications capabilities in a device is unlikely to be an end in itself. The end users are generally not as interested in the technology as in what the technology allows them to do.

Traditional telecommunications relies on voice communications as the single application of the technology, and this approach has been successful in the marketplace. As data communications services have become more widely available, there is increasing pressure to provide services that take advantage of those data capabilities.

The Wireless Application Protocol Forum was formed to create a standards-based framework, in which value-added data services can be deployed, ensuring some degree of interoperability.

2.2 USAGE CASES

The unique quality of Bluetooth, for the purposes of delivering value-added services, is the limited range of the communications link. Devices that incorporate Bluetooth are ideally suited for the receipt of location-dependent services. The following are examples of how the WAP client / server model can be applied to Bluetooth usage cases.

2.2.1 Briefcase Trick

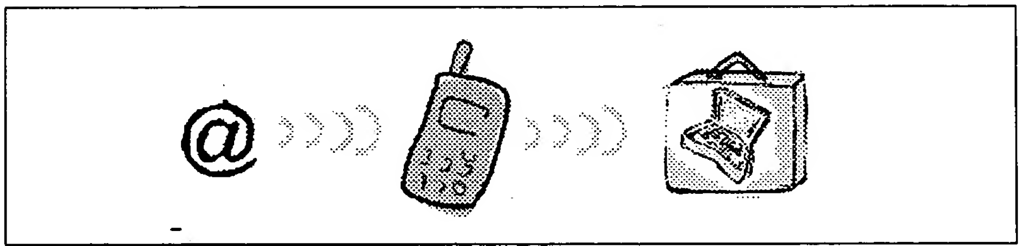

Figure 2.1: The 'Briefcase Trick' Hidden Computing Scenario

The Briefcase Trick usage case allows the user's laptop and mobile phone to communicate, without user intervention, in order to update the user's e-mail. The user can review the received messages from the handset, all without removing the laptop from its storage in a briefcase.

2.2.2 Forbidden Message

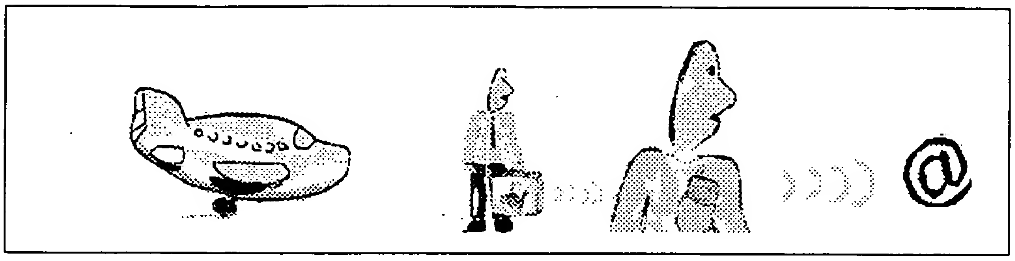

Figure 2.2: The 'Forbidden Message' Hidden Computing Scenario

The Forbidden Message usage case is similar to the briefcase trick. The user can compose messages in an environment where no dial-up connection is possible. At a later time the laptop wakes up, and checks the mobile phone to see if it is possible to send the pending messages. If the communications link is present, then the mail is transmitted.

2.2.3 WAP Smart Kiosk

The WAP Smart Kiosk usage case allows a user to connect a mobile PC or handheld device to communicate with a kiosk in a public location. The kiosk can provide information to the device that is specific to the user's location. For example, information on flights and gates in an airport, store locations in a shopping centre, or train schedules or destination information on a railway platform.

3 WAP SERVICES OVERVIEW

The Wireless Application Protocol is designed to provide Internet and Internet-like access to devices that are constrained in one or more ways. Limited communications bandwidth, memory, processing power, display capabilities and input devices are all factors driving the development of WAP. Although some devices may only exhibit some of the above constraints, WAP can still provide substantial benefit for those devices as well.

The WAP environment typically consists of three types of device: the WAP Client device, the WAP Proxy/gateway and WAP Server. In some cases the WAP Proxy/gateway may also include the server functionality.

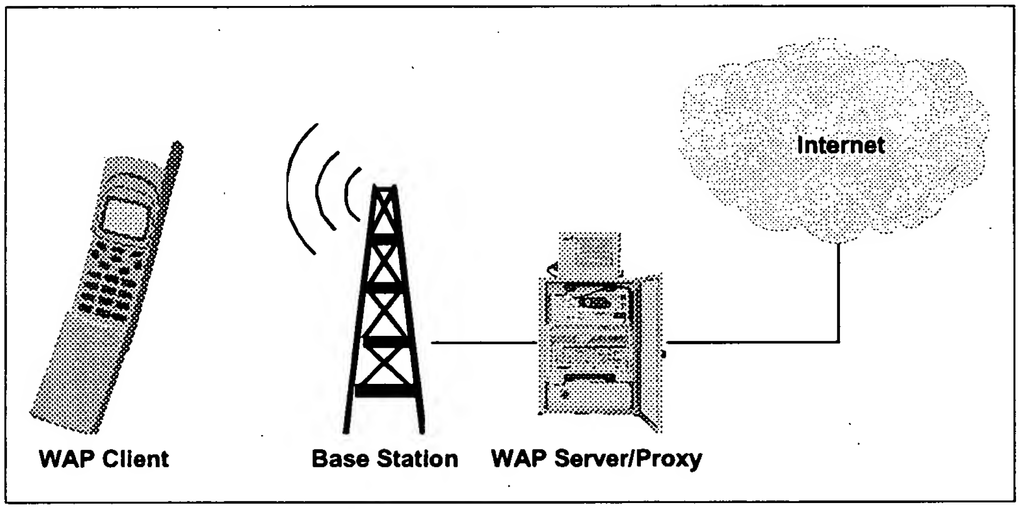

Figure 3.1: Typical WAP Environment

3.1 WAP ENTITIES

3.1.1 WAP Client

The WAP Client device is usually found in the hands of the end user. This device can be as powerful as a portable computer, or as compact as a mobile phone. The essential feature of the client is the presence of some type of display and some type of input device.

The WAP Client is typically connected to a WAP Proxy/gateway through a wireless network. (Figure 3.2 on page 503) This network may be based on any available technology. The WAP protocols allow the network to exhibit low reliability and high latency without interruption in service.

3.1.2 WAP Proxy/Gateway

The WAP Proxy/gateway acts as an interface between the wireless network, and the larger Internet. The primary functions of the proxy are to provide DNS name resolution services to WAP client devices and translation of Internet protocols and content formats to their WAP equivalents.

3.1.3 WAP Server

The WAP Server performs a function that is similar to a server in the Internet world. In fact, the WAP server is often an HTTP server. The server exists as a storage location for information that the user can access. This 'content' may include text, graphics, and even scripts that allow the client device to perform processing on behalf of the server.

The WAP Server logic may exist on the same physical device as the Proxy/gateway, or it may reside anywhere in the network that is reachable from the Proxy/gateway.

The server may fill the role of an HTTP server, a WSP server, or both.

3.2 WAP PROTOCOLS

The WAP environment consists of a layered protocol stack that is used to isolate the user agents from the details of the communications network. Figure 4.1 on page 506 illustrates the general architecture of the WAP protocol stack. Bluetooth will provide an additional data bearer service, appearing at the bottom of this diagram.

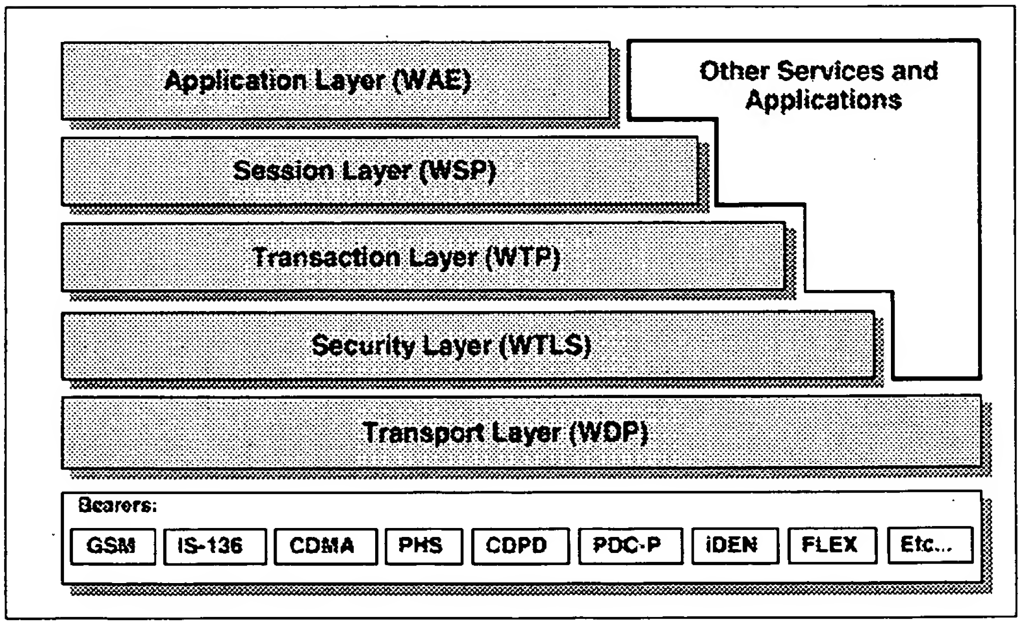

Figure 3.2: WAP Protocol Stack

3.2.1 Wireless Datagram Protocol (WDP)

The WDP layer provides a service interface that behaves as a socket-based UDP implementation. For a bearer service based on IP, then this layer is UDP. For bearer which do not provide a UDP service interface, then an implementation of WDP must be provided to act as an adaptation layer to allow socket-based UDP datagrams over the native bearer.

3.2.2 Wireless Transaction Protocol (WTP)

The WTP layer provides a reliable datagram service on top of the WDP (UDP) layer below.

3.2.3 Wireless Transport Layer Security (WTLS)

The WTLS layer is an optional component of the protocol stack that provides a secure data pipe between a client WSP session and its peer server WSP session. In the current version of the WAP specification, this session will terminate at the WAP server. There is currently a proposal before the WAP Forum for a proxy protocol, which will allow the intermediate WAP proxy to pass WTLS traffic across the proxy/gateway without decrypting the data stream.

3.2.4 Wireless Session Protocol (WSP)

The WSP layer establishes a relationship between the client application, and the WAP server. This session is relatively long-lived and able to survive service interruptions. The WSP uses the services of the WTP for reliable transport to the destination proxy/gateway.

3.3 CONTRASTING WAP AND INTERNET PROTOCOLS

The intent and implementation of the WAP protocol stack has many parallels with those of the Internet Engineering Task Force (IETF). The primary objective of the WAP Forum has been to make Internet content available to devices that are constrained in ways that make Internet protocols unsuitable for deployment.

This section compares the roles of the WAP protocol stack's layers with those of the IETF.

3.3.1 UDP/WDP

At the most basic layer, WAP and Internet protocols are the same. The WAP stack uses the model of a socket-based datagram (UDP) service as its transport interface.

Some Internet protocols also use the UDP service, but most actually use a connection-oriented stream protocol (TCP).

3.3.2 WTP/TCP

The wireless transport protocol (WTP) provides services that, in some respects, fill the same requirements as TCP. The Internet Transmission Control Protocol (TCP) provides a reliable, connection-oriented, character-stream protocol that is based on IP services. In contrast, WTP provides both reliable and unreliable, one-way and reliable two-way message transports. The transport is optimized for WAP's 'short request, long response' dialogue characteristic. WTP also provides message concatenation to reduce the number of messages transferred.

3.3.3 WTLS/SSL

The Wireless Transport Layer Security (WTLS) is derived from the Secure Sockets Layer (SSL) specification. As such, it performs the same authentication and encryption services as SSL.

3.3.4 WSP/HTTP

Session services in WAP are provided by the Wireless Session Protocol (WSP). This protocol incorporates the semantics and functionality of HTTP 1.1, while adding support for long-lived sessions, data push, suspend and resume. Additionally, the protocol uses compact encoding methods to adapt to narrowband communications channels.

3.3.5 WML/HTML

The markup language used by WAP is a compact implementation that is similar to HTML, but optimized for use in hand-held devices. WML is an XML-defined markup language.

3.3.6 WMLScript/JavaScript

WAP also incorporates a scripting language that is similar to JavaScript, but adapted to the types of constrained devices that WAP is targeted for.

4 WAP IN THE BLUETOOTH PICONET

In many ways, Bluetooth can be used like other wireless networks with regard to WAP. Bluetooth can be used to provide a bearer for transporting data between the WAP Client and its adjacent WAP Server.

Additionally, Bluetooth's *ad hoc* nature provides capabilities that are exploited uniquely by the WAP protocols.

4.1 WAP SERVER COMMUNICATIONS

The traditional form of WAP communications involves a client device that communicates with a Server/Proxy device using the WAP protocols. In this case the Bluetooth medium is expected to provide a bearer service as specified by the WAP architecture.

4.1.1 Initiation by the Client Device

When a WAP client is actively 'listening' for available Bluetooth devices, it can discover the presence of a WAP server using Bluetooth's Service Discovery Protocol.

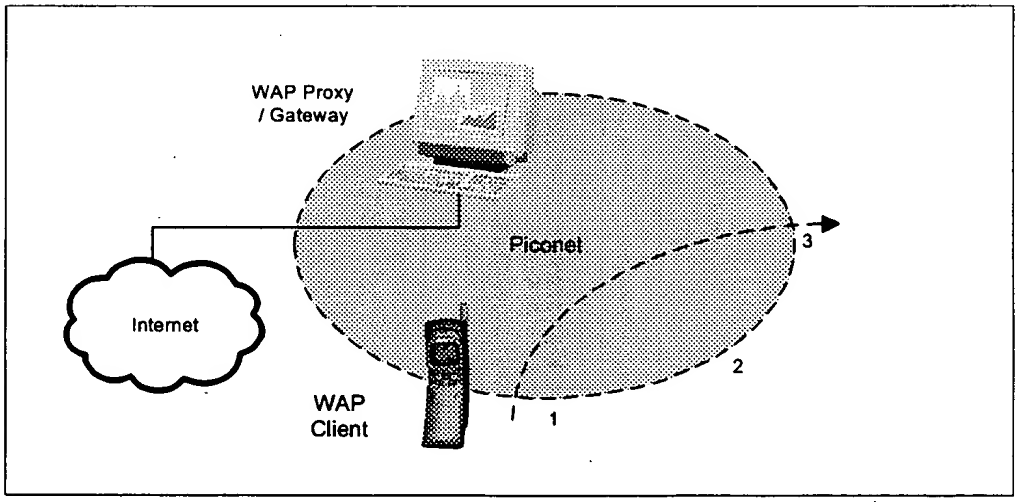

Figure 4.1: WAP Server / Proxy in Piconet

In Figure 4.1, stage 1 the WAP Client device is moving into range of the WAP Proxy/gateway's piconet. When the client detects the presence of the WAP proxy/gateway, it can automatically, or at the client's request, connect to the server.

4.1.1.1 Discovery of Services

The client must be able to determine the specific nature of the WAP proxy/ gateway that it has detected. It is expected that the Bluetooth Service Discovery Protocol will be used to learn the following information about the server:

- Server Name – this is a user readable descriptive name for the server.
- Server Home Page Document Name – this is the home page URL for the server. This is optional.
- Server/Proxy Capability – indicates if the device is a WAP content server, a Proxy or both. If the device is a Proxy, it must be able to resolve URLs that are not local to the Server/Proxy device.

In Figure 4.1, stage 2, the device is communicating with the WAP proxy/gateway. All WAP data services normally available are possible.

4.1.2 Termination by the Client Device

In Figure 4.1, stage 3, the device is exiting the piconet. When the device detects that communication has been lost with the WAP proxy/gateway, it may optionally decide to resume communications using the information obtained at discovery.

For example, a client device that supports alternate bearers may query the alternate address information of the server when that capability is indicated. The information should be cached for later access because the client device may leave the piconet at any time, and that information will no longer be available.

In the WAP Smart Kiosk example above, if the user wishes to continue receiving information while out of Bluetooth range, the Kiosk would provide an Internet address to the client device. When Bluetooth communications are not possible, the device could use cellular packet data to resume the client-server session.

This capability is implementation-dependent, and is provided here for illustrative purposes only.

4.1.3 Initiation by the Server Device

An alternative method of initiating communications between a client and server is for the server to periodically check for available client devices. When the server device discovers a client that indicates that it has WAP Client capability, the server may optionally connect and push data to the client.

The client device has the option of ignoring pushed data at the end user's discretion.

4.1.3.1 Discovery of Services

Through the Bluetooth Service Discovery Protocol, the server can determine the following information about the client:

- Client Name – this is a friendly format name that describes the client device
- Client capabilities – this information allows the server to determine basic information regarding the client's Bluetooth-specific capabilities

4.2 IMPLEMENTATION OF WAP FOR BLUETOOTH

In order to effectively implement support for WAP over Bluetooth, certain capabilities must be considered.

4.2.1 WDP Management Entity

Associated with an instance of the WDP layer in the WAP Protocol Stack is an entity that is responsible for managing the services provided by that layer. The WDP Management Entity (WDP-ME) acts as an out-of-band mechanism for controlling the protocol stack.

4.2.1.1 Asynchronous Notifications

The WDP-ME will need to be able to generate asynchronous notifications to the application layer when certain events occur. Example notifications are:

- New Client Node Detected
- New Server Node Detected
- Client Node Signal Lost
- Server Node Signal Lost
- Server Push Detected (detected as unsolicited content)

Platform support for these events is implementation-specific. All of the listed events may be derived through the Bluetooth Host Controller Interface (page 517), with the exception of Server Push.

4.2.1.2 Alternate Bearers

An implementation of WAP on a particular device may choose to support multiple bearers. Methods of performing bearer selection are beyond the scope of this document. The procedure to be followed is implementation-dependent. See Section 4.1.2 above.

4.2.2 Addressing

Two basic types of addressing are being used in the WAP environment: User Addressing and Proxy/gateway Addressing. User addressing describes the location of objects within the network, and is independent of the underlying bearer. Proxy/Gateway Addressing describes the location of the WAP proxy/ gateway that the device is communicating with. Proxy/Gateway addressing is dependent on the bearer type.

The end user deals mainly with Uniform Resource Locators (URL). These addresses are text strings that describe the document that is being accessed. Typically, the Proxy/gateway in conjunction with Internet Domain Name.

Servers resolve these strings into network addresses.

The address of the WAP Proxy/gateway is usually a static value that is configured by the user or network operator. When the user enters a URL, the request is forwarded to the configured WAP proxy/gateway. If the URL is within the domain of a co-located server, then it indicates that the document is actually WAP content. If the URL is outside of the WAP proxy/gateway's domain, then the WAP Proxy/gateway typically uses DNS name resolution to determine the IP address of the server on which the document resides.

The client device would first identify a proxy/gateway that is reachable through Bluetooth, then it would use the service discovery protocol to present the user with a server name or description. When the user selects a server, then the WAP client downloads the home page of the server (as determined by the discovery process; see section 4.1.1.1 on page 507) Once the user has navigated to the home page of the desired server, then all subsequent URLs are relative to this home page. This scenario presumes that the WAP Proxy/gateway and WAP Content server are all co-located in the Bluetooth device, although this structure is not required for interoperability.

A WAP Proxy/gateway/Server will typically provide a default URL containing the home page content for the server. A proxy-only device typically provides no URL or associated content.

4.3 NETWORK SUPPORT FOR WAP

The following specifies a protocol stack, which may be used below the WAP components. Support for other protocol stack configurations is optional, and must be indicated through the Bluetooth Service Discovery Protocol.

4.3.1 PPP/RFCOMM

Devices that support Bluetooth as a bearer for WAP services using PPP provide the following protocol stack support:

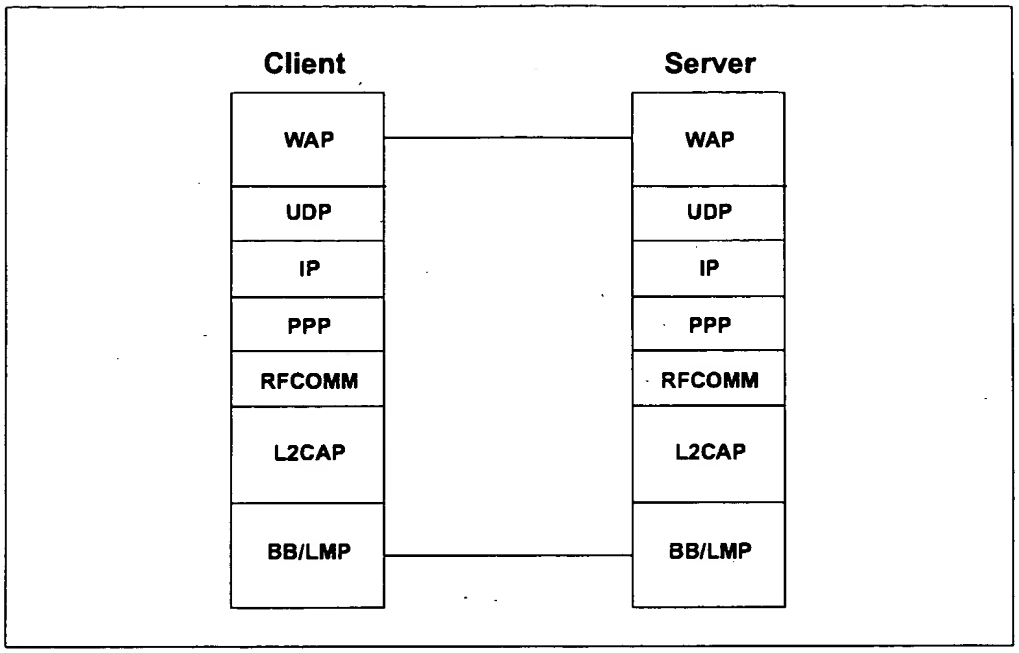

Figure 4.2: Protocol Support for WAP

For the purposes of interoperability, this document assumes that a WAP client conforms to the role of Data Terminal as defined in LAN Access Profile using PPP [6]. Additionally, the WAP server or proxy device is assumed to conform to the role of the LAN Access Point defined in [6].

The Baseband (page 33), LMP (page 185) and L2CAP (page 245) are the OSI layer 1 and 2 Bluetooth protocols. RFCOMM (page 385) is the Bluetooth adaptation of GSM TS 07.10 [1]. SDP (page 323) is the Bluetooth Service Discovery Protocol.

PPP is the IETF Point-to-Point Protocol [3]. WAP is the Wireless Application Protocol stack and application environment [5].

5 INTEROPERABILITY REQUIREMENTS

5.1 STAGE 1 – BASIC INTEROPERABILITY

Stage 1 interoperability for WAP over Bluetooth (all mandatory):

- Provide WAP Class A device compliance [7]
- Provide, through service discovery mechanisms, the network address for devices that support WAP proxy/gateway functionality.

5.2 STAGE 2 – ADVANCED INTEROPERABILITY

Stage 2 interoperability for WAP over Bluetooth (mandatory):

- All Stage 1 interoperability requirements are supported
- Provide Server Name and information about Server/Proxy capabilities through service discovery.
- Provide Client Name and information about Client Capabilities through service discovery.
- Asynchronous Notifications for Server.
- Asynchronous Notifications for Client.

6 SERVICE DISCOVERY

6.1 SDP SERVICE RECORDS

Service records are provided as a mechanism through which WAP client devices and proxy/gateways become aware of each other dynamically. This usage differs from other WAP bearers in that the relationship between the two devices will be transitory. That is, a Bluetooth device will not have a bearer-specific address configured or provisioned to a specific proxy/gateway.

Clients and proxy/gateways become aware of each other as they come in proximity of one another. The Bluetooth Service Discovery Protocol allows the devices to query the capabilities of each other as listed in the Interoperability Requirements section of this document.

Table 6.1 shows the service record for the WAP Proxy/gateway device.

Item	Definition	Type	Value	AttrID	Req
ServiceClassIDList				0x0001	M
ServiceClass0	WAP Proxy/Gateway	UUID	WAP		M
BluetoothProfile DescriptorList					M
ProfileDescriptor0				0x0009	M
Profile	Supported Profile	UUID	LANAccess UsingPPP [4]		M
Version	Profile Version	Uint16	*(varies)*		M
Protocol DescriptorList					O
Descriptor0	UDP	UUID	UDP		O
Parameter0	WSP Connectionless Session Port No.	Uint16	9200 *(default)*		O
Parameter1	WTP Session Port No.	Uint16	9201 *(default)*		O
Parameter2	WSP Secure Connectionless Port No.	Uint16	9202 *(default)*		O
Parameter3	WTP Secure Session Port No.	Uint16	9203 *(default)*		O
Parameter4	WAP vCard Port No.	Uint16	9204 *(default)*		O
Parameter5	WAP vCal Port No.	Uint16	9205 *(default)*		O
Parameter6	WAP vCard Secure Port No.	Uint16	9206 *(default)*		O
Parameter7	WAP vCal Secure Port No.	Uint16	9207 *(default)*		O

Table 6.1: Service Record format for WAP Proxy/Gateway devices

Item	Definition	Type	Value	AttrID	Req
ServiceName	Displayable Text name	String	*(varies, e.g. 'Airport information')*		
NetworkAddress	IP Network Address of Server	Uint32	(varies)		M
WAPGateway*	Indicates if device is origin server or proxy	Uint8	0x01 = Origin Server; 0x02 = Proxy; 0x03 = Origin Server and Proxy		M
HomePageURL	URL of home page document	URL			C1†

Table 6.1: Service Record format for WAP Proxy/Gateway devices

*. Stage 2 interoperability requirements.

†. If this parameter is omitted, then a default is assumed for origin servers as: http://*networkaddress*/index.wml

Item	Definition	Type	Value	AttrID	Req
ServiceClassIDList				0x0001	M
ServiceClass0	WAP Client	UUID	WAP_CLIENT		M
BluetoothProfile DescriptorList					M
ProfileDescriptor0				0x0009	M
Profile	Supported Profile	UUID	LANAccess UsingPPP [4]		M
Version	Profile Version	Uint16	*(varies)*		M
ServiceName	Displayable Text name of client	String	*(varies)*		O

Table 6.2: Service Record format for WAP Client devices

6.2 SDP PROTOCOL DATA UNITS

Table 6.3 shows the specified SDP PDUs (Protocol Data Units), which are required for WAP Interoperability.

PDU No.	SDP PDU	Ability to Send		Ability to Retrieve	
		WAP Client	WAP Proxy	WAP Client	WAP Proxy
1	SdpErrorResponse	M	M	M	M
2	SdpServiceSearchAttributeRequest	M	O	M	M
3	SdpServiceSearchAttributeResponse	M	M	M	M

Table 6.3: SDP PDU:s

6.3 SERVICE DISCOVERY PROCEDURE

In the simplest form, the signaling can be like this:

WAP Client or Proxy		WAP Client or Proxy
	SdpServiceSearchAttributeRequest ======================>	
	SdpServiceSearchAttributeResponse <======================	

WAP service discovery procedures are symmetrical. Each device must be able to handle all of the PDUs without regard for the current device role. A minimal implementation must return the service name string.

7 REFERENCES

[1] TS 101 369 (GSM 07.10) version 6.2.0

[2] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 50, RFC 1661, Daydreamer, July 1994.

[3] Simpson, W., Editor, "PPP in HDLC Framing", STD 51, RFC 1662, Daydreamer, July 1994.

[4] See Appendix VIII, "Bluetooth Assigned Numbers" on page 1009

[5] Wireless Application Protocol Forum, "Wireless Application Protocol", version 1.0, 1998

[6] Bluetooth Special Interest Group, "Bluetooth LAN Access Profile using PPP"

[7] Wireless Application Protocol Forum, "WAP Conformance", Draft version 27 May 1998